

Orientador

Professor Doutor Paulo Nazareno Maia Sampaio.

Professor Auxiliar do Departamento de Matemática e Engenharia da Universidade da Madeira

ABSTRACT

Generalized hyper competitiveness in the world markets has determined the need to offer better products to potential and actual clients in order to mark an advantage from other competitors. To ensure the production of an adequate product, enterprises need to work on the efficiency and efficacy of their business processes (BPs) by means of the construction of Interactive Information Systems (IISs, including Interactive Multimedia Documents) so that they are processed more fluidly and correctly.

The construction of the correct IIS is a major task that can only be successful if the needs from every intervenient are taken into account. Their requirements must be defined with precision, extensively analyzed and consequently the system must be accurately designed in order to minimize implementation problems so that the IIS is produced on schedule and with the fewer mistakes as possible.

The main contribution of this thesis is the proposal of *Goals*, a software (engineering) construction process which aims at defining the tasks to be carried out in order to develop software. This process defines the stakeholders, the artifacts, and the techniques that should be applied to achieve correctness of the IIS. Complementarily, this process suggests two methodologies to be applied in the initial phases of the lifecycle of the Software Engineering process: *Process Use Cases* for the phase of requirements, and; *MultiGoals* for the phases of analysis and design.

Process Use Cases is a UML-based (Unified Modeling Language), goal-driven and *use case* oriented methodology for the definition of functional requirements. It uses an information oriented strategy in order to identify BPs while constructing the enterprise's information structure, and finalizes with the identification of *use cases* within the design of these BPs. This approach provides a useful tool for both activities of Business Process Management and Software Engineering.

MultiGoals is a UML-based, *use case*-driven and architectural centric methodology for the analysis and design of IISs with support for Multimedia. It proposes the analysis of user *tasks* as the basis of the design of the: (i) user interface; (ii) the system behavior that is modeled by means of patterns which can combine Multimedia and standard information, and; (iii) the database and media contents.

This thesis makes the theoretic presentation of these approaches accompanied with examples from a real project which provide the necessary support for the understanding of the used techniques.

KEYWORDS

Software Engineering

User-Centered Design

Interactive Information Systems

Multimedia Authoring

Business Process Management

Unified Modeling Language

Methodologies and Techniques

User Interface (Screen) Design

Prototyping

RESUMO

A hiper concorrência generalizada dos mercados mundiais tem determinado a necessidade de oferecer melhores produtos aos actuais e potenciais clientes de forma a ganhar vantagem face à concorrência. Para garantir um produto adequado, as empresas precisam de trabalhar a eficiência e eficácia dos seus processos de negócio (PN) através da construção de Sistemas Interactivos de Informação (SII, incluindo Documentos Multimédia Interactivos) para que estes sejam processados de forma mais fluida e correcta.

A construção de um SII correcto é uma tarefa importante que só terá sucesso se as necessidades de cada interveniente forem tomadas em conta. Os requisitos têm que ser definidos com precisão, profundamente analisados e consequentemente o sistema tem que ser desenhado com exactidão de forma a minimizar problemas de implementação para que o SII seja produzido dentro do prazo e com o mínimo número de erros possível.

A principal contribuição desta tese é a proposta de *Goals*, um processo de construção de software que tem por objectivo definir as tarefas a serem realizadas para o desenvolvimento de software. Este processo define os participantes (stakeholders), os artefactos e as técnicas que devem ser aplicadas para atingir a correcção (correctness) do SII. Como Complemento este processo sugere a aplicação de duas metodologias na fase inicial do ciclo de vida de Engenharia de Software: *Process Use Cases* para a fase de requisitos, e; *MultiGoals* para as fases de análise e desenho.

Process Use Cases é uma metodologia baseada em UML (Unified Modeling Language), orientada aos objectivos (goals), e orientada aos casos de utilização para a definição de requisitos funcionais. Esta metodologia usa uma estratégia orientada à informação para identificar os processos de negócio e ao mesmo tempo construir a estrutura de informação da empresa, e finaliza com a identificação de casos de utilização no desenho dos processos de negócio. Esta “ferramenta” é útil para as actividades de Gestão de Processos de Negócio e para a Engenharia de Software.

MultiGoals é uma metodologia baseada em UML derivada de casos de utilização e centrada em arquitectura (architectural centric) para a análise de SIIs com suporte para multimédia. Propõe a análise das tarefas do utilizador como a base para o desenho de: (i) interface do utilizador; (ii) comportamento do sistema que é modelado

através de padrões que combinam multimédia e informação standard, e; (iii) base de dados e conteúdos multimédia.

Esta tese faz uma apresentação teórica destas abordagens acompanhada com exemplos de um projecto real que fornece o suporte necessário para a compreensão das técnicas usadas.

PALAVRAS CHAVE

Engenharia de Software

Desenho Centrado no Utilizador

Sistemas Interactivos de Informação

Autoria Multimédia

Gestão de Processos de Negócio

Unified Modeling Language

Metodologias e Técnicas

Desenho de Interface (Ecrã) do Utilizador

Protótipos

ACRONYMS

BP – Business Process
BPM – Business Process Management
CAP – Canonical Abstract Prototype
CASE – Computer Assisted Software Engineering
CTT – Concur Task Trees
DBMS – Database Management System
FR – Functional Requirement
Goals – Goal-Oriented Approach (led) Software
HCI – Human-Computer Interaction
HCM – Human-Centered Multimedia
HDM – Hypermedia Design Model
IIS – Interactive Information System
IMD – Interactive Multimedia Document
IS – Interaction Space
LUCID – Logical User Centered Interaction Design
NA – Norm Analysis
NFR – Non-Functional Requirement
OMG – Object Management Group
OO – Object Oriented
OOA – Object Oriented Analysis
OVID – Object, View and Interaction Design
PUC – Process Use Cases
QoE – Quality of Experience
QoS – Quality of Service
RE – Requirements Engineering
ROI – Return of Investment
RUP – Rational Unified Process

SE – Software Engineering

SQL – Structured Query Language

TPS – Transaction Processing System

UCD – Usage-Centered Design

UML – Unified Modeling Language

UWE – UML-based Web Engineering

VoIP – Voice over IP

Wisdom – Whitewater Interactive System Development with Object Models

ACKNOWLEDGEMENTS

I want to thank my Professor Paulo Sampaio who taught me how to make science and write papers and this thesis, for the numerous hours spent with me in the elaboration of the contributions here presented, and for always giving me support to go further.

I want to thank my Mother Margarida Dionísio and my friend Cabral without whom I would never have finished this thesis.

I want to thank my Father and my Grandmother, and to my entire family that always believed in me.

I want to thank my colleagues/friends Filipe Freitas and Paulo Vieira, and to my uncle João Dionísio for their participation in this work ;)

Also thanks to my cousin Cristina Ferraz for proofreading this thesis.

And thanks to my friend Jorge Gonçalves for treating the image of the front cover.

I finally want to thank all my friends (including my work colleagues) that in one way or another always encouraged me to finish this work.

TABLE OF CONTENTS

I. Introduction.....	21
I.1. Motivation.....	23
I.2. Problematic.....	24
I.3. Contributions.....	25
I.3.1. Requirements Phase.....	26
I.3.2. Analysis Phase.....	27
I.3.3. Design Phase.....	27
I.4. Organization.....	29
II. State of the Art: Interactive Information Systems and Multimedia Modeling.....	31
II.1. Introduction.....	32
II.2. Interactive Information Systems Architecture.....	34
II.2.1. Presentation Tier.....	34
II.2.2. Business Logic Tier.....	35
II.2.3. Data Tier.....	35
II.3. Multimedia Conceptual Approach.....	36
II.3.1. Multimedia Systems.....	36
II.3.2. Multimedia Applications.....	36
II.3.3. Interactive Multimedia Documents.....	37
II.3.4. Hypermedia Documents.....	39
II.4. Base Techniques.....	40
II.4.1. Requirements Definition.....	40
II.4.2. Requirements Definition Base Techniques.....	42
II.4.3. Analysis and Design.....	47
II.4.4. Analysis and Design Base Techniques.....	48
II.5. Related Works.....	54
II.5.1. Requirements.....	54
II.5.2. Analysis and Design.....	59
II.6. Conclusions.....	69

III. Requirements (Process Use Cases)	71
III.1. Introduction	72
III.2. Process Use Cases: An Overview	73
III.3. A Project	75
III.4. The Steps of Process Use Cases	76
III.4.1. Step 1 - Interiorize Project	76
III.4.2. Step 2 - Information Identification.....	77
III.4.3. Step 3 - Business Processes Identification.....	79
III.4.4. Step 4 - Use Cases Identification	83
III.5. Conclusions.....	87
IV. Analysis & Design (MultiGoals)	89
IV.1. Introduction	90
IV.2. MultiGoals: An Overview.....	91
IV.3. An Application	93
IV.4. The Steps of MultiGoals.....	94
IV.4.1. Step 1 - Use Cases.....	94
IV.4.2. Step 2 - Activity Diagram (Interaction Spaces + Tasks)	96
IV.4.3. Step 3 - Interaction Model (Task + System Responsibility).....	98
IV.4.4. Step 4 - Navigational Model (Interaction Spaces).....	100
IV.4.5. Step 5 - Presentation Model (Interaction Spaces + Tasks)	102
IV.4.6. Step 6 - Application Domain Model (Entities)	104
IV.4.7. Step 7 - Application Object Model (Entities Objects)	107
IV.4.8. Step 8 - Conceptual Model (System Responsibilities + Source).....	109
IV.4.9. Step 9 - System Behavior Model (System Responsibilities).....	111
IV.4.10. Step 10 - Temporal Model.....	115
IV.4.11. Step 11 - Application Architecture.....	117
IV.5. Conclusions.....	118
V. Case Study (MultiGoals)	119
V.1. Introduction	120
V.2. Training	122
V.3. Produced Diagrams	124
V.3.1. Step 1 - Use Cases.....	124
V.3.2. Step 2 - Activity Diagram.....	125
V.3.3. Step 3 - Interaction Model	128
V.3.4. Step 5 - Presentation Model	131
V.4. Conclusions.....	139

VI. Conclusions	141
VI.1. General Conclusions.....	142
VI.1.1. Requirements (Process Use Cases).....	142
VI.1.2. Analysis and Design (MultiGoals).....	143
VI.2. Future Work	146
VI.2.1. Requirements (Process Use Cases).....	146
VI.2.2. Analysis and Design (MultiGoals).....	147
List of Publications	149
References	150
Appendix A: Static and Run-Time Patterns	155
Appendix B: Stereotypes	161

LIST OF FIGURES

Figure 1: <i>Goals'</i> (partial view) defined phases.	26
Figure 2: <i>Goals'</i> (partial view) undefined phases	26
Figure 3: Sommerville's RE lifecycle.	41
Figure 4: <i>Wisdom's</i> Requirements Workflow.	44
Figure 5: <i>Business Process Model</i> .	45
Figure 6: Participation Map for a retail selling situation.	46
Figure 7: Activity-Task Map (partial) for retail selling.	46
Figure 8: <i>Wisdom's</i> Architecture.	48
Figure 9: Example of the application of an <i>essential use case</i> .	50
Figure 10: CAPs abstract tools and abstract materials.	51
Figure 11: Example of the application of CAPs.	51
Figure 12: <i>Concur Task Trees's</i> operators and types of <i>tasks</i> .	52
Figure 13: Example of the application of the <i>Concur Task Trees</i> .	52
Figure 14: Gonzalez' sequence of models.	54
Figure 15: Gonzalez' "IT Infrastructure" sequence of diagrams until reaching <i>use cases</i> .	55
Figure 16: Two <i>Norm Analysis</i> sentences.	56
Figure 17: Shishkov's derivation of <i>use cases</i> from norms.	56
Figure 18: Dijkman's mappings between business processes and <i>use cases model</i> .	57

Figure 19: Štolfa's sequential (a) and optional (b) patterns for derivation of <i>use cases</i> from business processes.	58
Figure 20: UWE's Conceptual Model.	61
Figure 21: UWE's Navigation Space Model.	61
Figure 22: UWE's Navigational Structure Model.	62
Figure 23: UWE's Presentation Model (Company).	62
Figure 24: UWE's Task Modeling.	63
Figure 25: W2000's Navigational Requirements Analysis'.	64
Figure 26: W2000's Hyperbase Information Design in-the-large(a) & in-the-small(b).	65
Figure 27: W2000's Hyperbase Navigation Design.	65
Figure 28: OMMMA's class diagram for the modeling of scenarios, applications and media classes.	66
Figure 29: OMMMA's interaction modeling.	67
Figure 30: OMMMA's Statechart diagram modeling the top level reactive behavior of the education application.	67
Figure 31: OMMMA's Sequence diagram modeling timed procedural behavior for the presentation of lecture discussion videos.	67
Figure 32: Business Process Management lifecycle.	73
Figure 33: <i>Process Use Cases's</i> BP.	74
Figure 34: Step 1- <i>High-level concept</i> for the project.	76
Figure 35: Step 2 - <i>Domain model</i> for the project.	78
Figure 36: <i>Business Process Model</i> for the "Obtain Recipe" business process.	80
Figure 37: <i>Business Process Model</i> for the "Make Event" business process.	81
Figure 38: <i>Business Process Model</i> for the "Advertise" business process	81
Figure 39: <i>Business Process Model</i> for the "Obtain Gastronomic Information" business process.	82
Figure 40: Step 3 – The <i>Business Process Model</i> for the project.	82
Figure 41: <i>Process use cases model</i> for "Obtain Recipe" business process.	84
Figure 42: <i>Process use cases model</i> for "Make Event" business process.	85
Figure 43: <i>Process use cases model</i> for "Advertise" business process.	85

Figure 44: <i>Process use cases model</i> for “Obtain Gastronomic Information” business process	86
Figure 45: Illustration of an interactive Multimedia scenario	93
Figure 46: Step 1: <i>Use Case</i> and complementary information for the example application.	94
Figure 47: Step 2 - <i>Activity diagram</i> for the example application.	97
Figure 48: Step 3 - <i>Interaction model</i> for the example application.	99
Figure 49: Step 4 - <i>Navigational model</i> for the example application.	101
Figure 50: Step 5 - <i>Presentation model</i> for the example application.	103
Figure 51: Step 6 - <i>Application Domain Model</i> for the example application.	105
Figure 52: Step 7 - <i>Object model</i> for the example application.	107
Figure 53: Step 8 - <i>Conceptual model</i> for the example application.	110
Figure 54: Sequence (of <i>system responsibilities</i>).	111
Figure 55: Parallel Fork.	112
Figure 56: Exclusive Fork.	112
Figure 57: Causal interruption generated from a user interaction.	113
Figure 58: Causal relation generated by the behavior of the system.	113
Figure 59: Step 9 - <i>System behavior model</i> for the example application.	113
Figure 60: Step 10 - <i>Temporal model</i> for the example application.	116
Figure 61: Step 11 - <i>Application architecture</i> for the example application.	117
Figure 62: Step 1 - <i>Use cases</i> of the Gastronomy Project.	124
Figure 63: Step 2 - <i>Activity diagram</i> for the "Catalog Recipe" <i>use case</i> .	125
Figure 64: Step 2 - <i>Activity diagram</i> for the "Advertise" <i>use case</i> .	127
Figure 65: Step 2 - <i>Activity diagram</i> for the "Obtain Gastronomic Information" <i>use case</i> .	127
Figure 66: Step 3 - <i>Interaction model</i> for the pair <i>task-system responsibility</i> “Select one of the Choices” - “Return the Chosen Category” (Recipe Situation).	129
Figure 67: Step 3 - <i>Interaction model</i> for the pair <i>task-system responsibility</i> “Select one of the Choices” - “Return the Chosen Category” (Search Situation)	130
Figure 68: Step 5 - "Recipe" IS versions 1 (blue ink) and 2 (red ink).	132

Figure 69: Step 5 - "Recipe" IS versions 2.1 and 3 (red ink, approved by the Client).	133
Figure 70: Step 5 - "Advertise" IS versions 1 (pencil) and 2 (red ink).	134
Figure 71: Step 5 - "News/Event/Publicity" IS version 2.1 (Approved by the Client).	134
Figure 72 : Step 5 - "HomePage" IS version 1.	135
Figure 73: Step 5 - "HomePage" IS versions 1.1 and 2 (red ink, approved by the Client).	136
Figure 74: Step 5 - "Advanced Search" IS version 1.	137
Figure 75: Step 5 - "Advanced Search" IS version 1.1.	138

LIST OF TABLES

Table 1: Requirements methodologies comparison	59
Table 2: Analysis and Design methodologies comparison.	68
Table 3: Steps of <i>Process Use Cases</i> methodology	74
Table 4: Steps of <i>MultiGoals</i> methodology	91
Table 5: <i>MultiGoals</i> training session schedule.	122
Table 6: Requirements methodologies comparison.	143
Table 7: Analysis and Design methodologies comparison.	144

I. INTRODUCTION

"It is a process (the Timeless Way of Building) which brings order out of nothing but ourselves. It cannot be attained, but it will happen of its own accord, if we will only let it. "

Christopher Alexander in The Timeless Way of Building
[Christopher Alexander, 1979]

Software construction is a major challenge. The requirements for the construction of software, such as time construction and budget restrictions do not allow great margins of error. For this reason, the phases that lead to the implementation of the software must be straightforward and precise regarding the production of artifacts that provide valuable information for the development of the software.

User-Centered Design has been successful in the task of understanding the user and consequently efficiently building more adequate software products. However, with the introduction of new technologies, there is the need of evolving the methods in a way that this extra complexity can be represented making these technologies useful for the development of better systems.

This thesis introduces *Goals*, a software construction process for the conception of correct software products for an enterprise aiming the resolution of specific information problems. *Goals* defines the phases (major activities), the *actors* (and their objectives), the outputs and inputs (artifacts), the triggers and the guidelines for each phase of the process which will ensure a higher rate of success for the project. For the

moment, the first 3 phases of a software development process are defined (requirements, analysis, and design).

Goals proposes two methodologies to be applied on these first three phases: (i) *Process Use Cases* for the requirements phase [Pedro Valente and Paulo Sampaio, 2007b] and; (ii) *MultiGoals* for the analysis and design phases [Pedro Valente and Paulo Sampaio, 2007a]. These methodologies provide a set of techniques that are applied to produce the necessary artifacts. *Process Use Cases* is a methodology that identifies *use cases* from the design of business processes, and *MultiGoals* is a methodology that fully designs the components of an Interactive Information System (with support for Multimedia) based on the analysis of *use cases* and associated *tasks*.

I.1. MOTIVATION

The precise and easy identification of functional requirements is crucial for the fluent development of a project. The identification of functional requirements (as *use cases*) in the design of the business processes (BPs) provides a tool that will allow profitable discussion between every stakeholder involved. Discussion will be based on the needed functionalities and, as a result, the reorganization of the BP will be clear to everyone.

Once functional requirements are identified, their analysis in terms of user *tasks* is facilitated since there is the previous knowledge of how these *tasks* can be carried out (during the execution of the BPs). Consequently, the design of the system will be carried out based on the context, increasing the probability of a more adequate and complete conception of the Interactive Information System (IIS).

IISs must be developed on schedule, as correct as possible and also have to be highly usable. For this reason, they must be designed in detail minimizing iterations between the phases of development and design. In order to produce a detailed design of the system, conceive its components (user interface, system behavior and information entities) and define the dependencies among these components, there is the need to use tools which detail the system up to a level when there are now doubts, for the future constructor (developer), on its functioning.

I.2. PROBLEMATIC

The construction of the correct software for an enterprise is a task which requires the use of all the resources efficiently. Human, physical and logical resources must be organized in such a way (in a project) that they will in the end bring an added value to every stakeholder involved. The accomplishment of such an organization will only be possible if every stakeholder is aware of its responsibility in the project and is able to negotiate its intentions in terms of project requirements.

In this negotiation, functional requirements assume a preponderant role regarding the final product of the project, the software, and the negotiation of the requirements will only be efficient if the objects under discussion are represented by a language that is understood by every intervenient. Usually, functionality and its associated implementation “effort” will be major issues, in which, artifacts like a *use case* model and an architectural view can make the difference between an abstract and a concrete discussion.

Technology provides each day more and more valuable solutions for the development of software, and from the combination of these new capabilities emerges complexity that needs to be represented. Multimedia is an example of useful technology that can bring an added value to traditional Interactive Information Systems. However, Multimedia requirements such as synchronization and user-system interaction need to be observed.

The modeling of systems is a highly elaborated task, for example a single user interface can easily have dozens of associated components. Hence, the modeler must be able to choose the models that will provide sufficient information about the design of the system so that developers will have no doubts about its implementation. For these reasons, analysis and design methodologies must be the more straightforward, complete and flexible as possible.

I.3. CONTRIBUTIONS

The main contribution of this thesis is the proposal *Goals*, which is a (business) process for the production of the correct Interactive Information System (IIS) for the resolution of a specific information problem. This process is defined into 6 different phases following a standard construction process: (i) requirements definition, (ii) analysis of the problem, (iii) design of the solution, (iv) development of the IIS, (v) test of the IIS and (vi) installation of the finished IIS. *Goals* also predicts that the software will need maintenance following two possibilities: (i) introduction of new requirements to the IIS, in which situation the complete process will be followed again, and, (ii) corrective maintenance in which case process is also executed from the beginning in order to identify where the mistake on the conception of the IIS was made.

Each phase of *Goals* is a business process itself in which a different methodology should be applied to produce information for the construction of the IIS. Although the *Goals* process is independent from the methodologies used, some restrictions should be observed in order to achieve the minimal quality for the global process and assure that full advantage is taken from the available inputs and that the needed outputs are also produced. Also, each phase defines: the human intervenient and their objectives, the minimal set of information inputs, and the outputs for the next phase.

The *Goals* process defines the phases of requirements, analysis and design (Figure 1). These phases are seen as the key for the success of the IIS producing the needed artifacts for the remaining phases (Figure 2) of development, test and installation. According to *Goals*, after the definition of the requirements, two other phases are applied for analysis and design of the software to be developed (or modified). For this reason, it is also an objective of this thesis to explain how requirements should be integrated with software analysis and how analysis should be integrated with design in order to achieve correct software definition.

Although all information generated along the process should be available to all the phases, *Goals* suggests sharing a minimal set of crucial information for correct system definition. The next sections describe the requirements, analysis and design phases illustrated in Figure 1.

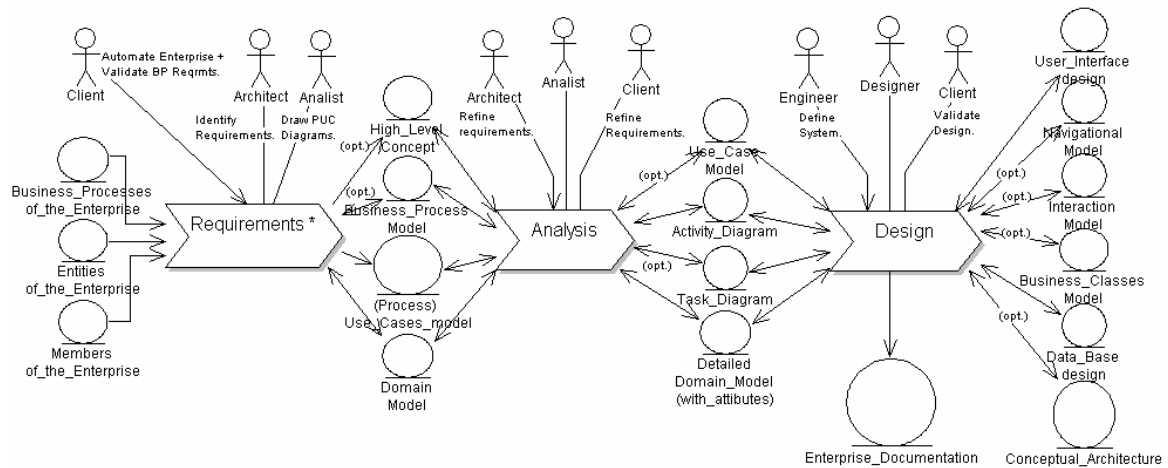


Figure 1: Goals' (partial view) defined phases.

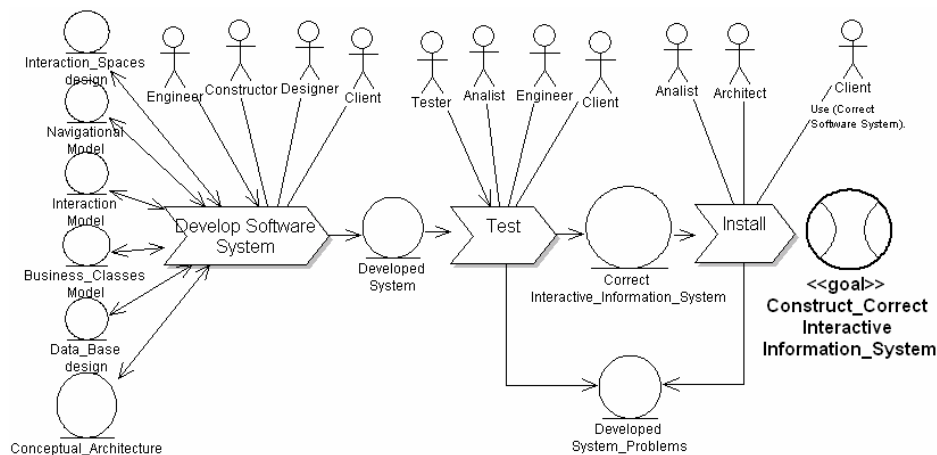


Figure 2: Goals' (partial view) undefined phases

I.3.1.Requirements Phase

The requirements phase aims at defining the requirements for the IIS. In this case the applied methodology is: (i) *use case-oriented*, in order to produce a *use case model* and (ii) *information-oriented*, in order to produce a *domain model*.

This phase that is triggered by the client with the intention of automating the enterprise regarding the resolution of some information problem, and can take advantage of artifacts that might already exist in the enterprise: business processes; information entities, and; members of the enterprise.

The following artifacts are defined as the minimal set of information to achieve functional requirements definition: (i) *domain model* - information entities of the enterprise, and; (ii) *use cases model* - the *use cases* of the system. Optionally a *high-level concept* and a *Business Process Model* can be elaborated.

In *Process Use Cases* architect, analyst and client work in order to produce the needed output elements: *high-level concept*; *Business Process Model*; (*process*) *use cases model*, and; *domain model*.

I.3.2. Analysis Phase

The analysis methodology was defined as: (i) object-oriented; (ii) *use case-driven*, and; (iii) architecture-centric in order to achieve consistency validation in system definition, i.e., to combine in one view usage, interaction interfaces, system behavior, information entities and the relations among them.

The following artifacts are defined as the minimal set of information to achieve comprehension of the problem: (i) an *activity diagram* - of the *use cases*, and; (ii) a *domain model* - detailed with attributes. Optionally a *use cases diagram* and a *task model* can be elaborated.

In *MultiGoals* architect, designer and client work to produce: (i) a *use cases model*; (ii) *activity diagrams*, and; (iii) an *interaction model*.

I.3.3. Design Phase

The choice for the design methodology should depend on: (i) the compatibility with the objects generated in the analysis phase; (ii) the non-functional requirements revealed in the analysis phase and the (iii) available resources, i.e., modeling detail needed for the development of the interactive system in: user interface usability, system behavior refinement and database integrity; the human resources available for the modeling, time and budget constraints.

The following artifacts are defined as the minimal set of information to achieve a solution: (i) user interface design, and; (ii) a database design. Optionally a *navigational model*, an *interaction model*, a business class model and a conceptual architecture can be elaborated.

In *MultiGoals* engineer, designer and client work as a team to produce: a (i) *navigational model*; (ii) *presentation model*; (iii) *application domain model*; (iv) *application object model*; (v) *conceptual model*; (vi) *system behavior model*; (vii) *temporal model*, and; (viii) a *multimedia architecture*.

To fully complement the *Goals* process the two methodologies already introduced are presented in this thesis to cover the defined phases of requirements, analysis and design. The first methodology is *Process Use Cases* which covers the phase of requirements, and begins with the specification of a statement that defines the project and ends with the identification of the *use cases* of the project. The second methodology is *MultiGoals* which covers the phases of analysis and design. This methodology takes advantage of the identified *use cases* and details them until the full definition of the system producing the outputs defined by *Goals*.

I.4. ORGANIZATION

This thesis is organized as follows:

- Chapter II presents the state of the art for the *use case*-oriented methodologies for the identification of requirements, and the UML-based methodologies for analysis and design of Interactive Information Systems with support for Multimedia;
- Chapter III presents the *Process Use Cases* methodology for requirements definition;
- Chapter IV presents the *MultiGoals* methodology for the analysis and design of Interactive Information Systems with support for Multimedia;
- Chapter V presents a case study of the modeling made by two Multimedia professionals with the *MultiGoals* methodology, and;
- Chapter VI presents some conclusions and future work.

II. STATE OF THE ART: INTERACTIVE INFORMATION SYSTEMS AND MULTIMEDIA MODELING

Software Engineering has largely benefited from the introduction of different technologies that combined have been providing useful ubiquitous solutions for the world in general. The relational database, the hypertext and programming languages are examples of tools that implement this reality.

From the generated complexity, solutions have emerged to defining architectural patterns, analysis and design techniques that are used with the intention of building better systems. This state of the art presents some of these existing solutions relative to Interactive Information Systems and Multimedia which represent the bridge that the contributions presented in this thesis aim to fulfill.

II.1. INTRODUCTION

The cost of a software project must have a comeback in terms of an adequate and usable system. Software Engineering (SE) has produced and gathered a number of different paradigms to help the construction of software systems involving techniques and tools that can be applied to define requirements, analyze the problem inherent to each requirement, design, implement and test the system.

Requirements Engineering (RE) has produced techniques to elicit and establish the requirements for a software project including the identification of *use cases* as a generally accepted technique to define functional requirements. The introduction of object-oriented (OO) analysis (OOA) (late 80's, 90's) has produced formal methods to analyze and model *use cases* using User-Centered Design techniques to understand the complexity of *tasks* users need to carry out on the system. The Human-Computer Interaction (HCI) and SE architecture-centric methods have produced OO techniques to design the (interactive information) system's user interface, code and data components in terms of a unique comprehensive structure. The establishment of techniques that cover the initial phases of software conception (requirements definition, analysis and design) has become a solution to model and consequently implement adequate and usable Interactive Information Systems (IISs).

Multimedia has become an usual solution to develop attractive applications in areas like education, arts, games and marketing. The initial challenges in the Multimedia history concerned issues like synchronization of media, definition of the structure of Multimedia applications or documents (separation of responsibilities of the code) and the production of Multimedia systems (including network and bandwidth, servers, protocols that assured quality of service) in order to provide services like video-on-demand, video-conferencing and online games. Although the same issues are always under development, recent Multimedia studies indicate that user satisfaction is gaining importance and issues like user interaction have crescent space in the Multimedia community, namely in Human-Centered Multimedia (HCM) in which the area of Multimedia interaction analyzes how people can interact with computerized systems in natural ways with special concerns on multimodal interaction (interaction based on multiple modes of interfacing) [Alejandro Jaimes et al., 2006].

This state of the art focus the efforts made in Software Engineering (SE) and Multimedia regarding the development of Interactive Information Systems (IISs). First we present an approach for architecture of IISs and how the responsibilities of the system can be divided in different tiers. After this we present an approach for architecture (structure) of an Interactive Multimedia Document (IMD) also dividing the responsibilities into different tiers (levels). Then we present the techniques that served as a basis for the contributions of this thesis: (i) requirements definition, and (ii) analysis and design of IISs. Finally we present the related works for both these areas.

II.2. INTERACTIVE INFORMATION SYSTEMS ARCHITECTURE

Interactive Information Systems (IISs) are developed to respond to the needs for automated information of an enterprise. An IIS is a set of components that collect, store, analyze and distribute structured information. IISs have evolved during time and can be divided into different categories: (i) operational level systems that appeared during the 50's where the transaction processing systems (TPSs) captured and processed the every day operations of the employees of the enterprise; (ii) management level systems that appeared during the 70's to support the enterprise's managers decision using techniques like data warehousing and data mining; (iii) knowledge level systems that appeared during this decade (2000's) to explicit, organize and distribute the knowledge within an enterprise.

With the growing complexity of IISs more powerful ways of structuring complexity are required [Rikard Land, 2002]. Software architectures have appeared as the way to control and document the components and relations among components of the implemented software in such a way that decisions can be made along the process of conceiving and evolving that IIS. The introduction of patterns has become a way to introduce standards in the architectural discussion. Architectural patterns like client-server architecture and object-oriented architecture provide an easy way to understand the parts involved on the decisions that need to be made to evolve a system.

In order to control the complexity of an IIS, responsibilities must be divided. Therefore, multi-tier architectures have become ubiquitous solutions, mainly 3-tier architectures [Robert Bretl et al., 1999] allow that the responsibilities of an IIS to be divided into user interface, business logic and data storage, that can be developed and maintained as independent modules. This assures traceability between code and architectural representation.

II.2.1. Presentation Tier

In a 3-tier architecture the presentation tier (1st tier) presents information (to the user) and supports application control (by the user). The presentation tier supports application control derived from the interactions generated by the user to perform his

tasks. When an interaction occurs the presentation tier triggers a request to the business logic tier which will provide the appropriate response involving navigation, data manipulation or both. The presentation tier invokes the functions provided by the business logic tier (2nd tier) returning or sending data in a previously agreed way and presenting it to the user. The 1st tier software usually runs on the client workstation under the perspective of a client-server architecture.

II.2.2. Business Logic Tier

The business logic (2nd tier) tier is responsible for receiving or sending data to the 1st tier (according to a previously agreed protocol), manipulating this data, and setting or collecting data from the 3rd tier (also according to a previously agreed protocol). The middle tier code typically carries out 3rd tier data queries, updates, and transactions to implement shared business logic. Data manipulation (performed by the IIS) is typically done on object representations of 3rd tier data. The configuration data used to set activities in the middle tier is usually stored in specialized files designed for specific configuration and management purposes.

II.2.3. Data Tier

The data tier (3rd tier) is responsible for receiving (and keeping) or returning (existing) data to the business logic tier (2nd tier) according to a previously agreed protocol that is usually based on structured query language (SQL) [Donald Chamberlin and Raymond Boyce, 1974]. SQL is a computer language designed for the retrieval and management of data in relational database management systems (DBMS), which, among other actions creates, updates and deletes records, creates and deletes databases, tables, views and stored procedures, and returns data based on complex queries.

Although other kinds of architectures would be eligible to fit our purpose of fully documenting IISs, the 3-tier architecture seems to be a balanced way to divide *system responsibilities* in such a way that they can be modeled separately promoting the reusability of the system components. The next section presents an analogous structure in a Multimedia perspective.

II.3. MULTIMEDIA CONCEPTUAL APPROACH

Multimedia is related to the synchronized presentation of different types of media objects where at least one of these objects is continuous (video, audio or animation). On the context of the work presented in this thesis, some basic concepts should be presented: (i) Multimedia systems; (ii) Multimedia applications; (iii) Interactive Multimedia Documents, and; (iv) Hypermedia documents.

II.3.1. Multimedia Systems

A Multimedia system is capable of managing the capture, generation, storage, recovery, processing, transmission and presentation of Multimedia information. Multimedia systems can be local or distributed.

Distributed Multimedia systems usually have large storage capacity and also require large bandwidth once they have to deal with jitter generated from e.g. hard disk access delays or codification. These systems should be the most tolerant to failures as possible.

II.3.2. Multimedia Applications

Multimedia applications are capable of handling (capturing, presenting and editing) Multimedia content and can be classified as presentational, conversational, and authoring Multimedia applications [Khalil Mehdi El-Khatib, 2005].

Presentational - are “one-way” Multimedia applications where (typically) Multimedia data is captured on one or more Multimedia servers and streamed to the receivers (users) over a broadband network. Receivers interact with the presentational Multimedia application by defining which data they want to receive and their preferences regarding the Quality of the Service (QoS). Examples of presentational Multimedia applications are news-on-demand, video-on-demand and distance education.

Conversational - are Multimedia applications where two or more users communicate with each other in real-time. The participants of a conversational application send and receive real-time data. Due to their interactive nature, conversational Multimedia

applications usually impose higher QoS requirements on all system components than presentational Multimedia applications. Examples of conversational Multimedia applications are voice-over-IP (VoIP) and video-conference.

Multimedia Authoring allow the implementation and presentation of interactive Multimedia documents (IMDs) and isolated media. Typically the Multimedia author can define: (i) the spatial features of the presentation, (ii) the temporal durations for the media and logical relations among media and (iii) the media storage. Well known examples of authoring tools are @Adobe Flash [Adobe, 2007a] and @3D Studio Max [Autodesk, 2007].

In general, Multimedia applications have a relation with interactive Multimedia documents once they have the capability of processing these documents.

II.3.3.Interactive Multimedia Documents

Interactive Multimedia Documents (IMDs) are digital documents composed by different types of media objects (image, text, graphics, video, audio, animation, digitalized sound or speech), integrated by means of temporal, spatial and logical relations, allowing user interaction and having at least one continuous media object (video, audio, animation, digitalized sound or speech).

In order to propose an approach for the modeling of applications that support Multimedia, it is indeed important to understand how IMDs can be structured. Thus, an IMD can be described according to a multi-level structure [Roberto Willrich, 1996]: (i) Presentation level, which describes how and where (spatial relations) each component of the document will be presented; (ii) Conceptual level, which describes the behavior of the IMD associated with the temporal and logical relations among the components of the document, and; (iii) Content level, which describes the information itself associated with each component of the document.

Besides, an authoring model should also consider the possible user interaction methods. Thus, the model should also describe anchors and links for Hypermedia navigation and other methods such as selection and data input. These structures are briefly discussed in the following sections.

Presentation Level

The Presentation level defines the spatial, temporal and sound characteristics for each component of the IMD. Thus, this level is composed by:

- The spatial characteristics of each visible component, such as the size, spatial position (absolute or relative to a virtual coordinates system) and presentation style;
- The temporal characteristics for the presentation of each dynamic component, such as the presentation speed, the initial and final position of presentation, and the number of possible repetitions;
- The characteristics related to sound are useful to define, for instance, the initial volume for the presentation of an audio sequence.

Conceptual Level

The Conceptual level is responsible for describing the components of a document and their logical and temporal relations:

- The components of a document are related to the description of the content of this document through a structure of modules with different semantics and granularity, for example: chapters, sections, sub-areas, etc.
- The logical structure of an IMD is related to the different possibilities a user has to navigate inside the document's structure. According to Ginige [Athula Ginige et al., 1995], three basic types of structures can be defined: linear, hierarchical and network. The choice of the most appropriate structure to an IMD depends upon the purpose of this document.
- The temporal relations are described based on events which can be produced during the presentation of an IMD. These events can be synchronous (when their occurrence can be predicted previously, such as the start or end of presentation of an image), or asynchronous (when their occurrence can not be predicted, such as the occurrence of a user interaction). The temporal relations describe not only the parallel and/or sequential presentation among media objects, but also the causal relations among them. In particular, the causal relations describe the conditional dependencies among the events associated with the components of an IMD. For instance, if a user interaction occurs over media object B, it interrupts the presentation of media object C.
- User interaction and subsequent system response is also defined at conceptual level. Thus, interactivity can be divided into the following categories: navigation, presentation control, environment control and information input.

Content Level

The Content level defines the information associated with each component of the IMD. Basically, the Content level describes the primitive data (image, text, graphics, video, audio, animation, digitalized sound or speech) related to each media object of the IMD. In this level, the information about access (URL), manipulation of primitive data and metadata shall be declared.

II.3.4. Hypermedia Documents

Hypermedia documents are a subclass of IMDs which support the integrated presentation of Multimedia and that implement the navigation among Multimedia contexts based on the concepts of node, anchors and hyperlinks.

The separation of responsibilities of the IMD (presentation, conceptual and content levels) can be equivalent to the responsibilities defined for the IIS (presentation, business logic and data tiers). This relation will be explored in chapter IV where the *MultiGoals* combines these levels and tiers into a single structure that supports the design of IISs with support for Multimedia. The following section presents the techniques that served as basis for the *Process Use Cases* and *MultiGoals* methodologies.

II.4. BASE TECHNIQUES

This section presents a conceptual approach on the activities for both requirements definition, analysis and design of Interactive Information Systems (IISs), and also describes the main techniques that served as basis for the methodologies presented in this thesis (*Process Use Cases* for requirements definition and *MultiGoals* for the Analysis and Design of IISs).

II.4.1. Requirements Definition

Requirements Engineering (RE) is the branch of Software Engineering (SE) that identifies the requirements for the implementation of an IIS. The conception of an IIS will only be successful if it supports all the needs of every stakeholder, i.e. individuals or organizations that win or loose with the success or failure of a system. From the diversity of stakeholders of a system different kinds of requirements are generated that need organization, conciliation and validation before the IIS is designed.

Requirements elicitation is the main objective of RE and is based on the identification of the stakeholders and their goals [Bashar Nuseibeh and Steve Easterbrook, 2000]. A number of elicitation techniques can be applied such as: Traditional techniques - use of questionnaires, interviews, and analysis of existing documentation; Group elicitation techniques - to foster stakeholder agreement to elicit a richer understanding of needs; and Prototyping - when there is a great deal of uncertainty about the requirements prototyping which can provoke discussion over concrete material.

According to Ian Sommerville in [Ian Sommerville, 2005], RE has a lifecycle cycle (Figure 3) composed by the activities of: Elicitation (identify sources of information and discover the requirements from them); Analysis (understand the requirements, their overlaps and their conflicts); Validation (check with the stakeholders if the requirements are what they really need); Negotiation (try to reconcile conflicting views and generate a consistent set of requirements); Documentation (write down the requirements in a way that every stakeholder understands); and Management (control the requirements changes).

According to the same author, in a competitive market where there is the need for rapid software delivery and the need to get improved return of investment (ROI), RE

activities should be integrated with the activities of system design and implementation in order to produce software in increments which represent an added value to the client.

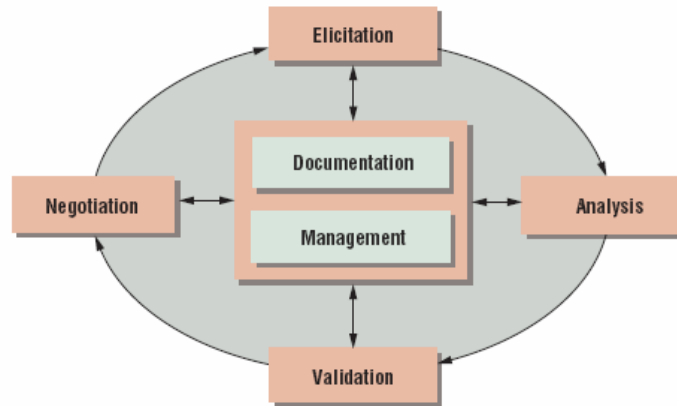


Figure 3: Sommerville's RE lifecycle.

Stakeholders related to the business management activities will be concerned (for example) over how the new system will improve the functioning of the enterprise, the budget of the project and the implementation time. Most times these non-functional requirements (NFRs) will dictate the success or failure of the IIS.

In opposite, users want the new system to improve the efficiency and efficacy of their work and will generate a set of functional requirements (FRs) that will have direct impact on the design of the IIS and which can collide with the existing NFRs. Indeed, the more functionality a system needs the more expensive it will be and more time it will take to be implemented. Budget, time constraints and FRs of the system will have to be conciliated in such a way that every stakeholder needs are sufficiently satisfied.

The decomposition of stakeholder's goals into different levels of abstraction leads us to the identification of *use cases*, which are the point where users interact with IISs in order to carry out useful *tasks*. *Use cases* are the most widely used technique in RE to express user requirements (user-centered development has adopted *use cases* as the cornerstone of its process) once they describe the *task* that the user carries out on the IIS and serve as guidance for the IT professionals that will implement the system.

The modeling of the enterprise business processes is required in order to understand the context of an IIS: the organizational structure, business rules, the goals and *tasks*. Business process modeling will help to understand how individuals from the organizational structure combine their efforts (respecting business rules) to achieve a certain business goal.

Modeling business process during RE will open a space for Business Process Management (BPM) to take place before the new IIS is implemented. A new IIS will have an impact in the enterprise and the BPM activities can measure that impact and predict how the new automated business process will benefit the enterprise's goals.

All these RE engineering tasks must be supported by RE techniques that ensure the final objective of requirements definition. The RE techniques that served as basis for one of the contributions of this thesis, *Process Use Cases*, a methodology for requirements definition are presented on the next section with a brief introduction to the methodology.

II.4.2. Requirements Definition Base Techniques

Process Use Cases (PUC) is the result of the need to easily identify *use cases* and relate them to the parts of the software implemented using a semantically understandable conceptual *architecture* model that gathers both *business processes* (BPs) and system components (and dependencies among them). The main goal of PUC is to develop, in a sequence of 4 steps, the *process use cases model*, in which *actors* and *use cases* [Larry Constantine, 2006] come together to achieve a first stage of functional requirements definition (the interactions between users and system, the *use cases*).

Different abstractions provided by different techniques are used to represent the information acquired within PUC. These techniques are: UML [Object Management Group, 2003]; *Wisdom* [Nuno Nunes, 2001]; the *High-Level Concept* [Charles Kreitzberg, 1999]; the *Business Process Model* [Hans-Erik Eriksson and Magnus Pencker, 2001] and *Usage-centered design* [Larry Constantine, 2006].

UML

The *Unified Modeling Language* (UML) [Object Management Group, 2003] is an object-oriented (OO) modeling language for specifying, visualizing, constructing and documenting the artifacts of software systems. UML version 0.9 was published in 1996 by Grady Booch, Jim Rumbaugh, and Ivar Jacobson as an attempt to normalize the semantics and notation of other existing OO languages. UML has become the standard modeling language in software industry and has been, from the late 90's, the reference to a number of other methodologies, notations and techniques that restrict or extend UML's models and notation.

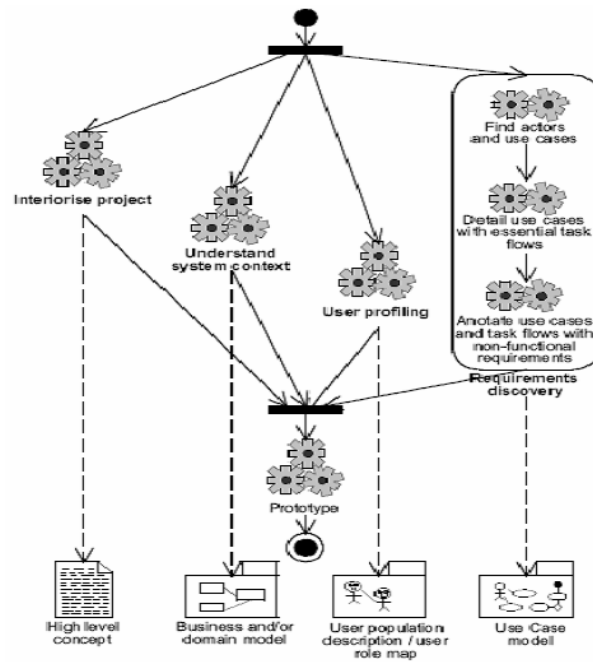
RUP [Philippe Kruchten, 1999], the *Rational Unified Process*, developed initially by the Rational Corporation, is the software development process that explains how to apply UML.

Besides the UML notation, *class diagram* and *activity diagram* are used within PUC to produce the *domain model* and *process use cases model* respectively.

Wisdom

Wisdom was proposed as a solution to bridge Usability Engineering and Software Engineering and as a way to apply Software Engineering in small software development companies [Nuno Nunes, 2001]. *Wisdom* is an evolutionary, prototyping, agile UML-based method which provides an activity dedicated to requirements definition (the Requirements Workflow) within its process.

The Requirements Workflow (Figure 4) starts with the “interiorize project activity” which is a short textual description that indicates what the system should and should not do, and what are the potential benefits and anticipated risks. The second activity is “understand system context” that produces a *domain model* (an UML class diagram) when the problem domain is very simple or when the development team is experienced in the domain. In addition, a business model (class diagram using the business process profile of the UML) and *activity diagrams* to describe the business processes should be elaborated when the problem is very complex or when there is little knowledge of the domain. The activity “User Profiling” produces a user role model to describe who are the users, how they are grouped and what are their salient characteristics. The last activity is “requirements discovery” that encompasses several sub-activities, they are (i) finding *actors* and *essential use cases*; (ii) detailing *essential use cases* with *activity diagrams*, and; (iii) annotating non-functional requirements to *use cases*.



The activities for requirements discovery (especially “process interiorization” and “requirements discovery”) defined in *Wisdom* provide the main concepts behind *Process Use Cases*. The concept of *entity* used in both *Process Use Cases* and *MultiGoals* is also provided by *Wisdom*.

High-Level Concept

The *Logical User Centered Interaction Design* (LUCID) was proposed as a way of describing the approach to interface design at Cognetics Corporation [Charles Kreitzberg, 1999] with the objective of improving software usability. LUCID is composed of 6 stages: (i) Envision - Develop UI (User Interface) Roadmap which defines the product concept, rationale, constraints and design objectives; (ii) Analyze - Analyze the user needs and develop requirements; (iii) Design - Create a design concept and implement a key screen prototype; (iv) Refine - Test the prototype for design problems and iteratively refine and expand the design; (v) Implement - Support implementation of the product making late stage design changes where required and Develop user support components, and; (vi) Support - Provide roll-out support as the product is deployed and gather data for next version.

The *High-Level Concept* is a statement defined within the LUCID Framework and is the first step for the envisioning of the product. The *High-Level Concept* is seen as a mission statement for a product to help focus product development. The same concept is used in *Process Use Cases*.

Business Process Model

The *Business Process Model* [Hans-Erik Eriksson and Magnus Pencker, 2001] is a notation developed by Hans-Erik Eriksson and Magnus Pencker as a way to help enterprises to model their business processes and their context using UML and ease the relation to the implementation of the enterprise information system (Figure 5).

After the business process is identified, the following information is associated: Inputs, Resources and Information (Resources serve as "inputs" and information "supply" information); Events (that trigger the business process); Outputs (may be a physical object, a transformation of raw resources or an overall business result), and; Goals (the reason for the existence of the business process).

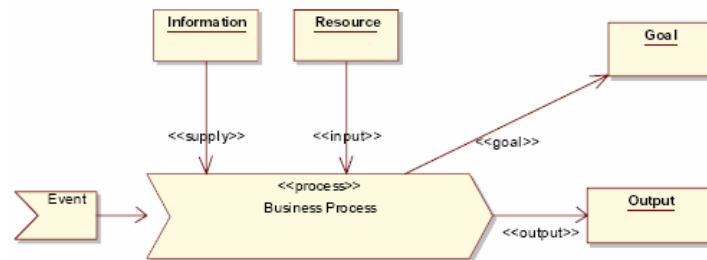


Figure 5: *Business Process Model*.

The *Business Process Model* provides the (adapted) notation used in *Process Use Cases* for modeling BPs and their interaction with users and information.

Usage-Centered Design

Usage-centered design (UCD) is a model-driven process for user interface and interaction design developed by Larry Constantine (Constantine & Lockwood, Ltd.) [Larry Constantine and Lucy Lockwood, 2000]. Since UCD has special concerns with usability, detailed attention is given to the *tasks* users need to carry out on the system to be developed, and, to the usage of their system. This has led to the definition of several basic concepts related to Human-Computer Interaction such as: (*essential*) *use cases*, *actors* and roles.

UCD requirements definition is based in activity theory which is a way of describing and characterizing the structure of human activity of all kinds, that was first introduced by Russian psychologists Rubinshtein, Leontiev, and Vigotsky in the early part of the XX century.

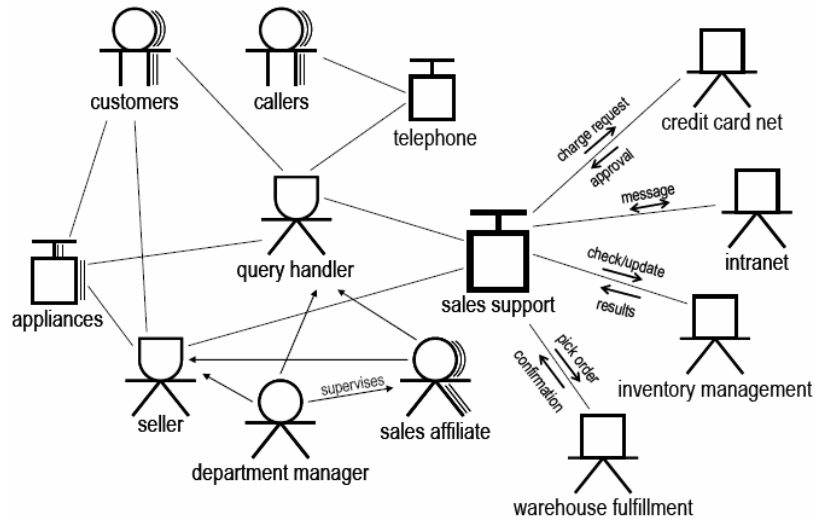


Figure 6: Participation Map for a retail selling situation.



Figure 7: Activity-Task Map (partial) for retail selling.

The following activities are carried out in a straightforward process: (i) Activity Map (Figure 7, upper part) – representation of the activities relevant to the design problem and the interrelationships among them; (ii) Activity Profiles - purpose, place and time, participation, and performance related to each relevant activity; (iii) Participation Map (Figure 6) – a representation of the participants (*actors*, roles, players, system actors) and their relationships with each other and with the various artifacts involved in the activity; (iv) Activity-Task Map (Figure 7) – *tasks* (user interactions with the system, *essential use case*) and actions (actions which are not carried out by the interaction with the system) are extracted and related to the activities previously identified in the activity map.

The UCD concept of (*essential*) *use case* is applied in *Process Use Cases*. Indeed, the notion of *use case* provided by UCD is seen as crucial for the correct identification of *use cases*.

These techniques are the foundation to the requirements contribution provided by this thesis. The conceptual approach for the analysis and design is presented on the next section.

II.4.3. Analysis and Design

Software is being developed in the world almost since the appearance of the first computer in the 40's. The potential provided by computers was an attractive solution to solve information problems for enterprises that manipulated large amounts of data and needed automation for their business processes.

Analysis and design for software-based Interactive Information Systems mainly for transaction processing systems (TPSs) in the 50's was initially developed without the use of any formal tools except for the use of flowcharts. With the proposal of new technologies and the need to meet expectations, and without the tools to correctly understand the problem to be solved and consequently design the appropriate software, the software industry has entered an age of crisis. Developed software had inappropriate functionality, was developed outside schedule and over the budget.

It was only in the 70's that the structured analysis and design was introduced by Yourdon and Constantine [Edward Yourdon and Larry Constantine, 1979] and only became generally accepted during the 80's, and that has established modeling as a fundamental activity in Software Engineering. But it was only in the late 80's that object oriented methods made their way in SE supporting the modeling of the IIS components as individual objects observing the relations of aggregation (or composition) and inheritance.

The introduction of both *use cases* and *task analysis* (also in the 80's) was a major breakthrough regarding the modeling of adequate user interfaces once they are a valuable tool to specify users, understand the context of use and define responsibilities of the system. As a complement (to user interface analysis and design), architectural-centric methods lead to the modeling of *system responsibilities* and data components as objects (and establish the relations among them) as a catalyst for reuse and easier system maintenance.

These analysis and design tasks must be supported by Software Engineering techniques that ensure the final objective of system design. The analysis and design techniques that served as basis for the second contribution of this thesis, *MultiGoals*, a methodology for the analysis and design of Interactive Information Systems with

Multimedia support are presented on the next section with a brief introduction to the methodology.

II.4.4. Analysis and Design Base Techniques

MultiGoals is the result of the need to design Interactive Information Systems (with support to Multimedia) comprehensively and in detail, specifying user interface objects, the correspondent *system responsibilities* and the data components of the system.

The main techniques applied by *MultiGoals* are UML [Object Management Group, 2003], *Wisdom* [Nuno Nunes, 2001], *Usage-centered design* [Larry Constantine, 2002] (which includes *essential use cases* [Larry Constantine and Lucy Lockwood, 2000] and *Canonical Abstract Prototypes* [Larry Constantine, 2003]) and *Concur Task Trees* [Fábio Paternò et al., 1997]. UML provides the basic notation of the methodology, *Wisdom* provides the main Software Engineering process, *Usage-centered design* provides specific techniques for requirements definition and user interface design, while *Concur Task Trees* provide the technique for user-task modeling.

UML

As presented in section II.4.2.

Wisdom

As presented previously, *Wisdom* [Nuno Nunes, 2001] was proposed as a solution to bridge usability engineering and Software Engineering, and, as a way to apply Software Engineering in Small Software Development Companies. *Wisdom* is an evolutionary, prototyping, UML-based method. The *Wisdom* method provides a tool to rapidly achieve a stage of implementation based on a few and easy to understand sequence of diagrams, effectively reducing the great quantity of models provided by UML and RUP, focusing on the essentials of the system being developed. The main diagrams proposed within *Wisdom* are illustrated in the Figure 8.

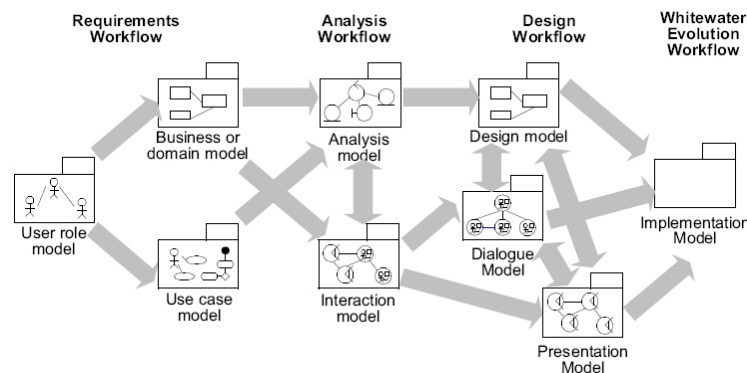


Figure 8: *Wisdom's* Architecture.

As a complement to the requirements workflow already presented in section II.4.2, *Wisdom* predicts the following activities until reaching an implementation model. The Analysis Workflow starts with the Internal System Analysis that encompasses the sub-activities of: (i) Identify General Analysis Classes - in which the classes captured in the requirements workflow execution are refined, with this objective *Wisdom* suggests that the CRC [Kent Beck and Ward Cunningham, 1989] method is applied as an effective way to extract analysis classes and corresponding responsibilities and (ii) Structure Analysis Classes - in which activity the analysis classes are then structured into analysis stereotypes and responsibilities distributed to build an internal architecture. The second (and concurrent) activity of the Analysis Workflow is the Interface Architecture Design activity that concerns the external architecture of the system which encompasses the sub-activities of: (i) identify and (ii) structure interaction classes for both *task* and *interaction spaces* classes. The final activity of the Analysis Workflow relates both internal and external architectures in a single architecture that ensures that the future design can be seamlessly built upon this structure of classes.

The Design Workflow starts with the Internal System Design which encompasses the sub-activities of: (i) Prioritizing and selecting candidate *use cases* for design and (ii) Design *use case* classes, in which the analysis classes are refined at both responsibility and association level integration non-functional requirements annotated in the requirements phase. The second (and concurrent) activity of the Design Workflow is the User Interface Design which encompasses the sub-activities of: for user *tasks* (i) Prioritize and select *tasks*; (ii) Refine *tasks*; and (iii) Define temporal relationships between *tasks*; and for *interaction spaces* (iv) Prioritize and select *interaction spaces*; (v) Identify contained *interaction spaces* where the complex *interaction spaces* are decomposed in different contained or navigable *interaction spaces*; (vi) Map actions to dialogue model in which an initial correspondence is established between *tasks* and *interaction spaces*; (vii) Map input and output elements to interface components, and; (viii) Relate *tasks* and *interaction spaces*. The final activity of the Design Workflow is to build a prototype of the system. *Wisdom* suggests that the Bridge method [Tom Dayton et al., 1998] part 2 to map *task* flows to *task* objects (classes) and part 3 to map *task* objects to the graphical user interface.

Wisdom provides the basics of a Software Engineering process for *MultiGoals*, especially the notions of *interaction space*, *user task*, *system responsibility* and *entity*.

Essential Use Cases

Usage-centered design (UCD) [Larry Constantine, 2002] is an object-oriented based approach for interactive system design that firstly applied the already existing *essential use cases* [Larry Constantine and Lucy Lockwood, 2000]. *Essential use cases* are an evolution of concrete *use cases* which are usually used in a large variety of scope, detail, focus, format, structure, style, and content by both software engineers and Interface Designers, that results in imprecision in the definition of the requirement.

By definition, an *essential use case* is a single, discrete, complete, meaningful, and well-defined *task* of interest to an external user in some specific role or roles in relationship to a system, comprising the user intentions and *system responsibilities* in the course of accomplishing that *task*, described in abstract, technology-free, implementation independent terms using the language of the application domain and of external users in role.

Essential use cases focuses in what the user really needs to accomplish and provide a way to connect the design of the user interface back to the essential purpose of the system and the work it supports, contributing to fulfill a gap between Software Engineering and interface design. An illustration of the application of *essential use cases* is depicted in Figure 9 where user and system “collaborate” to accomplish a test. The user intentions are the *tasks* that the user might want to carry out on the system instead of the traditional approach where the user has to take the actions technically desirable influenced by the available technology resulting in more complex systems to use.

User Intentions
System Responsibilities
Asynchronous Extensions
optionally at point do {resetting test}
optionally at any point do {reporting fatal exception}
1. present list of standard test setups
2. select standard test setup
3. display selected test setup
4. optionally [do modifying test setup]
5. confirm test setup
6. run test as set up and report
7. optionally [print test results]
8. print and confirm

Figure 9: Example of the application of an *essential use case*.

The *use cases* used in *MultiGoals* are *essential use cases*.

Canonical Abstract Prototypes

Canonical Abstract Prototypes (CAPs) [Larry Constantine, 2003] are part of the *Usage-centered design* [Larry Constantine, 2002]. CAPs allow the modeling of a complete set of user interactions that can occur in the components of the user interface and also the position, size, layout, and composition of the user interface features. The user interface is modeled by the combination of abstract tools and abstract materials.

The use of CAPs can largely contribute for the better and faster understanding of the functionality of the user interface, especially if a software development team exists. Some of the most commonly used CAPs notations are depicted in Figure 10 and in Figure 11 is presented an example of the application of CAPs in which the user is able to navigate among items of a list.

















SYMBOL	INTERACTIVE FUNCTION	EXAMPLES
(Abstract Tools)		
	action/operation*	Print symbol table, Color selected shape
	start/go/to	Begin consistency check, Confirm purchase
	stop/end/complete	Finish inspection session, Interrupt test
	select	Group member picker, Object selector
	create	New customer, Blank slide
	delete, erase	Break connection line, Clear form
	modify	Change shipping address, Edit client details
	move	Put into address list, Move up/down
	duplicate	Copy address, Duplicate slide
	perform (& return)	Object formatting, Set print layout
	toggle	Bold on/off, Encrypted mode
	view	Show file details, Switch to summary
(Abstract Materials)		
	container*	Configuration holder, Employee history
	element	Customer ID, Product thumbnail image
	collection	Personal addresses, Electrical Components
	notification	Email delivery failure, Controller status

Figure 10: CAPs abstract tools and abstract materials.



Figure 11: Example of the application of CAPs.

CAPs are used in *MultiGoals* to complementarily describe user interface interaction and functionality.

Concur Task Trees

Concur Task Trees (CTTs) [Fábio Paternò et al., 1997] is a notation, proposed by Fabio Paternò for *task* modeling which is a central and familiar concept in Human-Computer Interaction.

A *task model* details user's goals and the strategies adopted to achieve those goals in terms of actions that the users perform, the objects involved in those actions, and the underlying sequencing of activities. CTTs are based in a graphical notation that supports the hierarchical structure of *tasks*, which can be interrelated through a powerful set of operators that describe the temporal relationships between *subtasks*.

The operators and the types of possible *tasks* are presented in Figure 12.

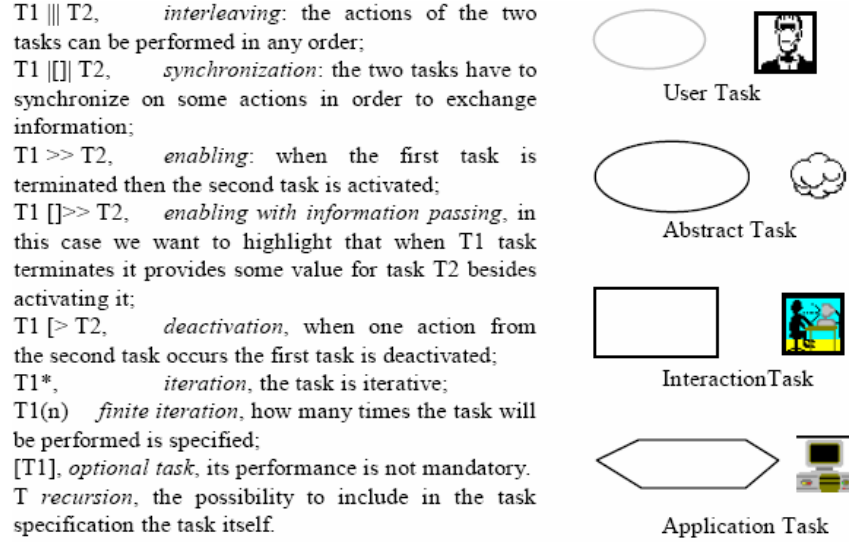


Figure 12: Concur Task Trees's operators and types of *tasks*.

An example of the application of CTTs is presented in Figure 13 where the "Application" *task* is decomposed in the "editing" *task* that deactivates the "close" *task*. The "editing" *task* is further detailed into the "specify" *task* that enables the "perform" *tasks* passing information to this second *task*.

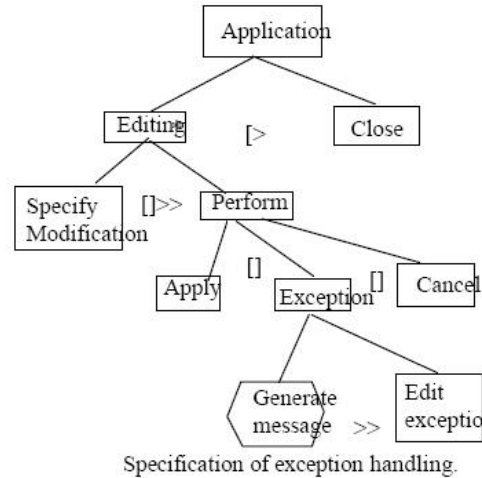


Figure 13: Example of the application of the Concur Task Trees.

CTTs were adapted to fit *MultiGoals* modeling, relating the *tasks* defined with the system behavior (the system response), however, the main guidance lines towards interface design defined in CTTs are preserved.

The previously described techniques differently contributed for the elaboration of *MultiGoals*. The next sections present works that are related to the contributions of this thesis regarding requirements definition, analysis and design of Interactive Information Systems.

II.5. RELATED WORKS

This section presents the related work for both requirements, analysis and design. Based on what is defined in the *Goals* process as requirements for the choice of methodologies for the first three phases of the Software Engineering process, the following was observed:

- Requirements – methodologies that identify *use cases* as a result of the analysis of business processes. The production of a *domain model* was not observed in order to enlarge the scope of our analysis;
- Analysis and design – UML-based *use case*-driven methodologies for the design of Interactive Information Systems with support for Multimedia. The production of a *domain model* was not observed in order to enlarge the scope of our analysis.

II.5.1. Requirements

Most approaches found in the literature argue that *use cases* should be identified as a result of the design of business processes and should be used to specify the requirements for a software project. We now briefly present these approaches and then proceed to their comparison.

Gonzalez

Gonzalez in [Jose González and Juan Sánchez Díaz, 2007] proposes an extensive approach (Figure 14) which defines “Business Strategy” by means of an organizational mission statement, the strategic goals that support the statement, the measures that indicate business success and their target measures. Afterwards, the “Business Infrastructure” is represented by the organizational operational structure through a process map, a role model, a resource model, and business processes.

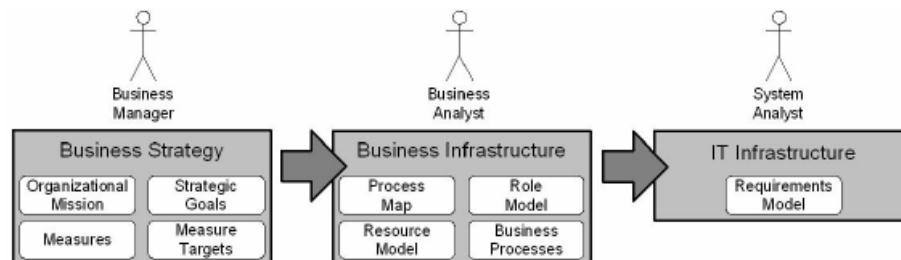


Figure 14: Gonzalez' sequence of models.

The process is completed by following the “IT Infrastructure” step (Figure 15) building a “business process goal tree” composed by “goals” and “tasks” that are derived from the “business process” and “resource model” diagrams by means of heuristic rules. The “business process goal tree” is then labeled according to the nature of their tasks and goals: (A) Automated goal; (M) Manual goal; (C) Ceased goal; or (IS) automatic goal. Finally, the *use cases model* is built based upon the A tasks for human intervention and the IS tasks for system intervention.



Figure 15: Gonzalez’ “IT Infrastructure” sequence of diagrams until reaching *use cases*.

This approach is very interesting regarding the analysis of the enterprises’ strategy and related goals which are major issues in Business Process Management. However, this approach does not provide a structured way to identify *use cases* from business processes once the “business process goal tree” (which is elaborated based on heuristic rules) represents a major drawback between the business processes design (no specific modeling technique is provided) and the *use cases* diagram design.

Usage-centered design

As previously presented in section II.4.2, *Usage-centered design* (UCD) comprehends 4 steps for requirements definition: (i) Activity Map – representation of the activities relevant to the design problem and the interrelationships among them; (ii) Activity Profiles – purpose, place and time, participation, and performance related to each relevant activity; (iii) Participation Map – a representation of the participants, their relationships with each other and with the various artifacts involved in the activity, and; (iv) Activity-Task Map – *tasks* and actions are extracted and related to the activities previously identified in the activity map.

UCD provides the notion *essential use case* that is used in our contribution. UCD’s approach is based on activity theory and provides a very interesting way of analyzing the user’s intervention within business processes modeling its interaction with existing artifacts and other solicitations. From this interaction, *tasks* and actions are derived.

Norm Analysis

Shishkov in [Boris Shishkov and Jan Dietz, 2005] proposes the modeling of business processes based on “Norm Analysis” (NA), a semantic tool that specifies the norms which are the (business) rules and patterns of behavior within an organization in a sentence with the following structure (Figure 16): whenever <condition> if <state> then <agent> is <deontic operator> to <action>.

<i>f-NORM 1</i>	<i>f-NORM 2</i>
Whenever	Whenever
<Client/Hotel has decided to use HRB>	<The subscription fee is paid by the Client/Hotel>
If <The Client/Hotel initiates subscription>	If <The Client initiates the match-making>
Then <the Client/Hotel>	Then <HRB>
Is <Obligated to>	Is <Obligated to>
To <Pay the subscription fee>	To <Perform the match-making>

Figure 16: Two Norm Analysis sentences.

After the NA sentences (norms) are specified, the *use cases* are identified based on the “actions” that need to be carried out by the intervenient of the norm. In Figure 17 the *use cases* “Arrange Subscription Payment” and “Perform Match-Making” are derived from the norms “f-NORM 1” and “f-NORM 2” respectively.

This approach is very interesting since it describes the business rules that the enterprise members and system must implement in order to run the business properly. However, this approach is very limited for the description of complex business processes that recurrently have more than one condition and one action (or activity).

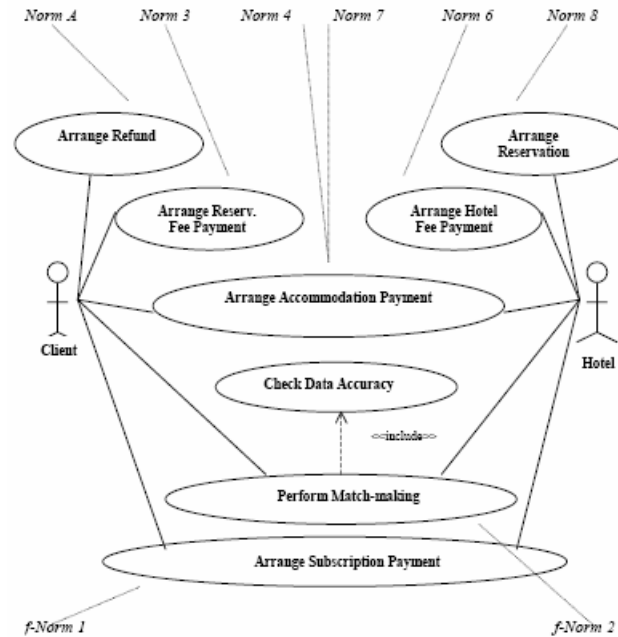


Figure 17: Shishkov’s derivation of *use cases* from norms.

Dijkman

Dijkman in [Remco Dijkman and Stef Joosten, 2002] proposes a detailed procedure to transform business process models into *use case* diagrams by mapping roles to *actors*, steps to *use cases*, and tasks to interactions, etc. as presented in Figure 18. A “Step” is a sequence of “Tasks”. Dijkman’s method consists in modeling the business processes using UML *activity diagrams*, making the mappings between the identified components, and consequently producing a *use cases model* as the final step.

In Dijkman’s approach the business process activities (“steps” in the Figure) are mapped directly into *use cases*, roles are mapped into *actors*, and sub-activities (“tasks” in the Figure) are mapped into interactions within *use cases*. Dijkman further defines the mapping of “guards in transitions” and “alternative paths through branches”.

Business Process Concept	Use Case Concept
Role	Actor
Step	Use Case
Association between Role and Step	Association between Actor and Use Case
Task	Interaction
Task in a Step	Interaction in a Use Case
Transition between Tasks in the same Step	Ordering between Interactions in the same Use Case
Guard on Transition	Constraint on Interaction
Alternative Path through a Branch	Alternative Path Description of a Use Case, or Extending Use Case

Figure 18: Dijkman’s mappings between business processes and *use cases model*.

Wisdom

As previously presented in section II.4.2, Wisdom comprehends 4 steps for requirements definition: (i) “interiorize project activity” producing a High-Level Concept; (ii) “understand system context” to produce a *domain model* and/or a business model; (iii) “user profiling” producing a Role Model, and; (iv) “requirements discovery” that encompasses finding *actors* and *essential use cases*, detailing *essential use cases* with *activity diagrams*, and annotating non-functional requirements to *use cases*.

Wisdom provides the main concepts behind our contribution, especially the activities for requirements discovery (“process interiorization” and “requirements discovery”).

Štolfa

Štolfa in [Svatopluk Štolfa and Ivo Vondrák, 2006] proposes that business processes are designed using *activity diagrams* and that a mapping is made between the activities

of the business process and *use cases*. The mapping can be “one-to-one” or “mapping several actions to *use cases*” by applying the “sequential” pattern or the “optional” pattern respectively.

In the “sequential” pattern several actions are mapped to one single general *use case* using “mapping several actions to one *use case*” method, and sequential actions (or whole parts of other patterns that may replace these actions) are mapped to other *use cases* that are included by the first one (Figure 19a). The “optional” (Figure 19b) pattern is applied when there is a condition block in the *activity diagram*. The condition block and the action are mapped to a single general *use case* and the action is mapped to a *use case* that extends the first one.

In this approach *use cases* are mapped “one-to-one”, however, in our perspective the grouping of *use cases* as predicted in the “sequential” and “optional” patterns adds extra complexity to the *use cases model* that is not necessary.

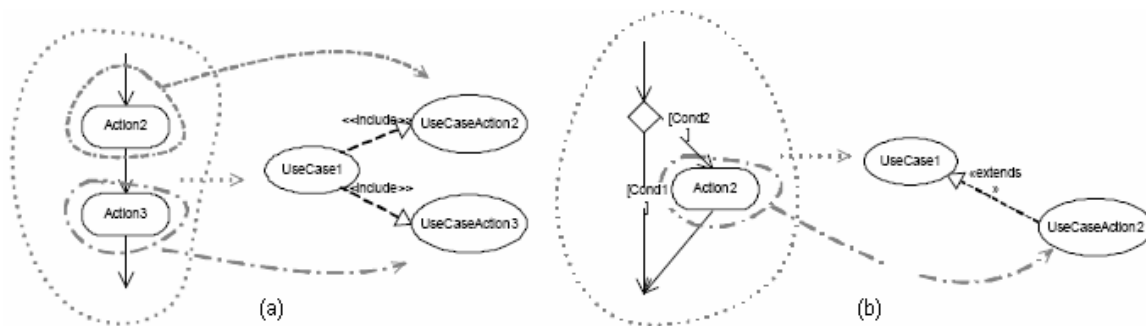


Figure 19: Štolfa's sequential (a) and optional (b) patterns for derivation of *use cases* from business processes.

Following the presentation of the related works for requirements definition, we present a summary of the diverse methods in Table 1 which includes the following criteria:

- Project Interiorization – (if the methodology includes) a way of promoting the understanding of the scope of the project for the project community;
- Business Strategy Description – description of the business strategy defined by an enterprise in the financial market that she is involved in;
- Data Modeling – modeling of a single structure of the information that is manipulated by the enterprise;
- Business Process Context – definition of the triggers of the business processes, their outputs and relations with other business processes;
- Goals Identification – identification of the goals of the enterprise and their relation with existing business processes;

- Business Process Design – design of the activities of the business processes and the *actors* that perform those activities;
- Business Rules – identification of the business rules defined within the functioning of the enterprise;
- Business Processes Resources– identification of the resources consumed and produced by the business processes;
- Use Cases Identification – identification of *use cases* as a result of the analysis of the business processes.

Table 1: Requirements methodologies comparison

	Gonzalez	UCD	NA	Dijkman	Wisdom	Štolfa
Project Interiorization					X	
Business Strategy Description	X					
Data Modeling	1	2			X	
Business Process Context		X			X	
Goals Identification	X					
Business Process Design	X	X	X	X	X	X
Business Rules		3	X			
Business Processes Resources	X	X				
Use Cases Identification	X	X	X	X	X	X

1 - Domain model of the used resources.

2 - Systems, Artifacts and Tools Identification.

3 - Included in the artifact design.

Table 1 makes the comparison of methodologies that both design business processes and identify *use cases*. By the analysis of the table, Gonzales, Wisdom and *Usage-centered design* are the more complete methodologies, and are the only three that provide data or system/artifact/tools modeling. Only Gonzalez' approach structures the business strategy of the enterprise and identifies business goals. The Wisdom approach is the only one that has concerns on the project interiorization. Only *Usage-centered design* and Wisdom design the context of the business processes. Moreover, only *Norm Analysis* and *Usage-centered design* include business rules in their work.

The previous table showed that sufficient approaches exist to extract requirements based on business processes covering all the criteria that were taken into account. Our approach (that is presented in Chapter III) tries to cover most of the criteria as possible while being simple to use and expressive.

II.5.2. Analysis and Design

User centered development has produced a large number of methodologies for the analysis and design of Interactive Information Systems (IISs). The application of object-oriented concepts to system modeling (in the 90's) and the introduction of *use cases* by Ivar Jacobson and colleagues encouraged the proposal of methodologies for the design of IISs. For instance, some of these methodologies are: *Usage-centered design*

[Larry Constantine, 2002]; *Wisdom* [Nuno Nunes, 2001]; *Idiom* [Mark Van Harmelen, 2001], and; OVID [Dave Robert et al., 1998]. Relatively to Multimedia there are a number of methodologies (over 15 were studied) for the modeling of interactive Multimedia documents (IMDs) with detailed interest in synchronization which however are not UML-based and for this reason were not included in our study. Some examples are: MING-I [Chung-Ming Huang et al., 2004]; ZYX [Susanne Boll and Wolfgang Klas, 2001], and; TOCPN [Kyoungro Yoon and Bruce Berra, 1998].

With the crescent impact of the internet, the establishment of UML as the standard information systems modeling language and the interest in the potentialities provided by Multimedia, a number of UML-based methodologies for the design of Hypermedia were conceived which are in the scope of our work. However, most of the existing approaches do not make the analysis of the user *tasks* by providing a *use case* model or alternatively user *task* analysis, focusing only in the conception of the domain, the user interface and navigation, and for this reason important Hypermedia contributions like OOHDM [Daniel Schwabe and Gustavo Rossi, 1998], WebML [Stefano Ceri et al., 2003] and NDT [Maria Escalona et al., 2003] were left out of our study. As a complement, OMMMA [Stefan Sauer and Gregor Engels, 2001], an approach for the design of Interactive Multimedia Documents, which is not *use case*-driven, was included in our study since it provides the modeling of synchronization and media content, features that are left out of the Hypermedia contributions. For each approach we provide an overview of the most important features ending with a comparison among the studied contributions.

UML-based Web Engineering

UML-based Web Engineering (UWE) [Nora Koch and Andreas Kraus, 2002] is a *use case*-driven methodology for the analysis and design of Hypermedia document that adapted the Unified Process to the Hypermedia conception. This methodology produces in three steps the following artifacts: (i) Conceptual Design - (that produces the) conceptual model; (ii) Navigational Design - navigation space model and navigational structure model, and; (iii) Presentational Design - presentation model.

The conceptual model is built taking into account the functional requirements captured with *use cases*, and traditional object-oriented techniques are used to construct the conceptual model, such as finding classes, defining inheritance structures and specifying constraints. The output will be a class diagram as presented in Figure 20.

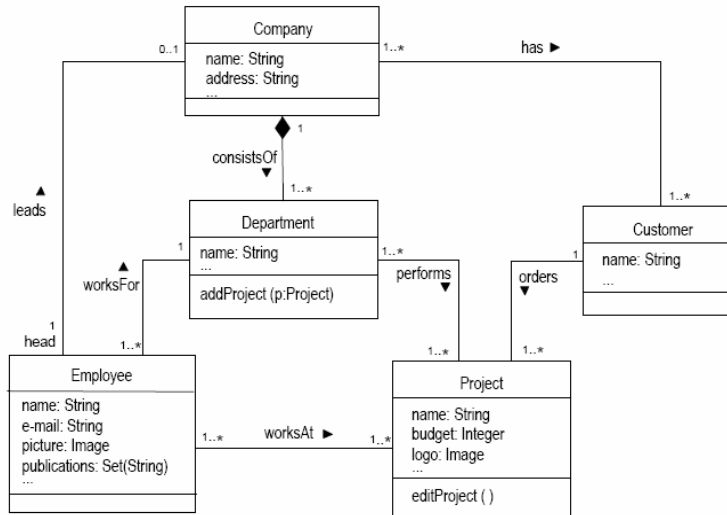


Figure 20: UWE's Conceptual Model.

The next step is the Navigation Space Model (Figure 21). The author defines which of the existing classes are “navigational classes”, i.e. classes whose instances are visited by the user during navigation. Complementarily, the author defines among which “navigational classes” will exist navigation, defining the “direct navigability” (among classes).

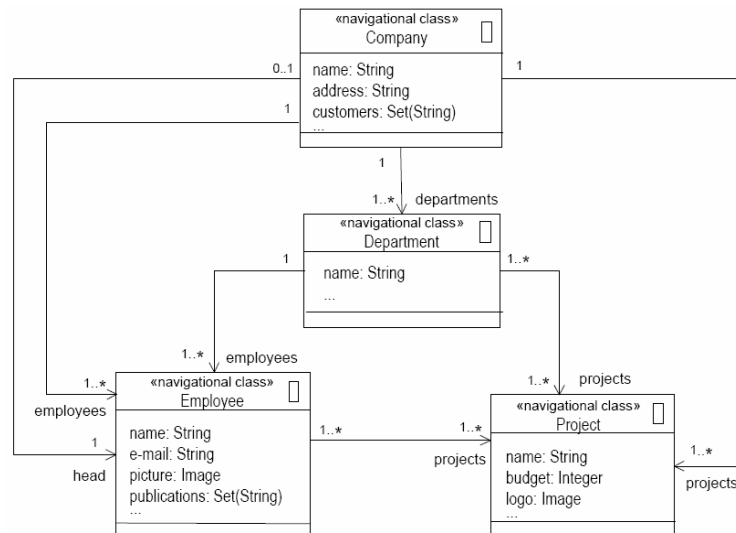


Figure 21: UWE's Navigation Space Model.

Once the Navigation Space Model is completed, each existing navigation will be enhanced by one “access element” (indexes, guided tours or queries) that defines how the navigation will take place. Complementarily, “menus” (and “menu items”) will be defined and will be aggregated to the existing “navigational classes”. Each “menu

item” has a name and owns a link either to an instance of a “navigational class” or to an “access element”. This diagram is called Navigational Structure Model (Figure 22).

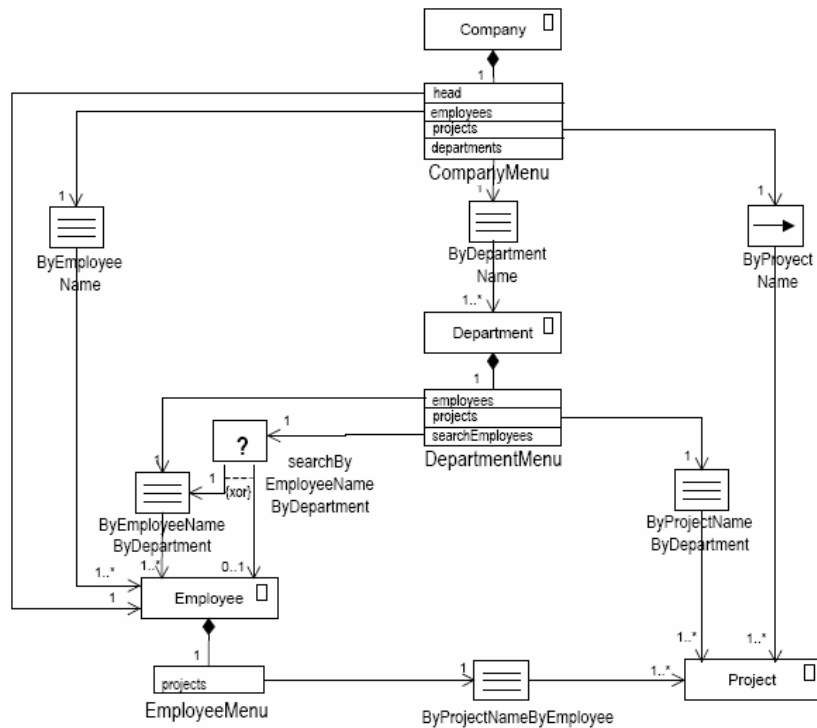


Figure 22: UWE's Navigational Structure Model.

The next step of the methodology is the Presentational Model which describes how the information within the “navigational classes” and the “access elements” are presented to the user. This is done by constructing an abstract interface design similar to a user interface sketch (Figure 23).

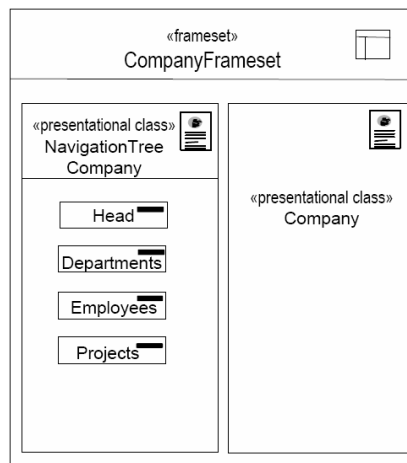


Figure 23: UWE's Presentation Model (Company).

It is only after the presentation model is produced that the UWE methodology makes the analysis of the user *tasks*. This analysis is carried out modeling the user *tasks* with *activity diagrams*. Each *task* is represented by an activity that is further refined (using

the “refine” stereotype”) into sub-activities. An example of the analysis of the user tasks is presented in Figure 24.

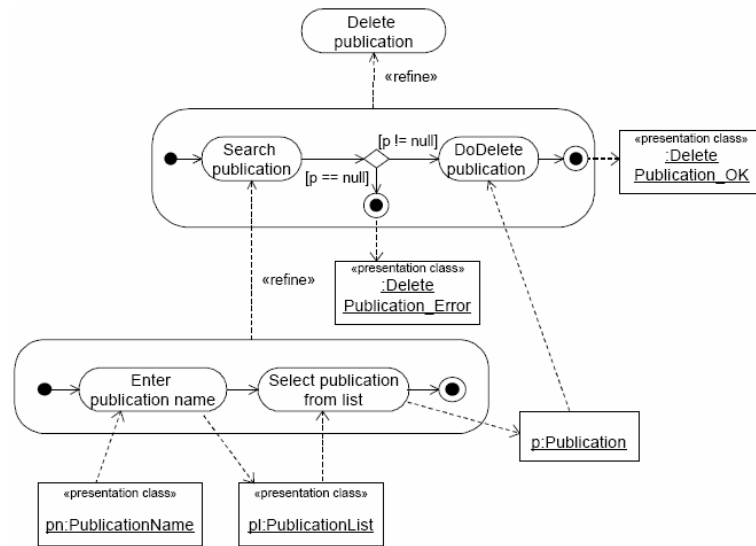


Figure 24: UWE's Task Modeling.

UWE models with accuracy the navigation and the presentation of information to the user based on a *domain model* of the structure of the information of the business under analysis. However, this approach makes the construction of the user interface dependent of the domain not taking advantage of the existence of the *use cases* and not giving sequence to the analysis of the user *tasks*. This can be considered a bottom-up approach since the interface depends on the domain. Contrarily, there is no modeling of the media requirements, and synchronization is left out of the scope of the methodology.

W2000

W2000 [Luciano Baresi et al., 2001] is the evolution of the Hypermedia Design Model (HDM) [Franca Garzotto et al., 1993] an Hypermedia methodology recognized as the ancestor of a family of several design methodologies. With the new version (W2000) the authors wanted to make the methodology UML-compliant and to adapt the old models to the new challenges of Hypermedia in the beginning of the 2000's, e.g. e-commerce.

The methodology starts with the “requirements analysis”, an activity that encompasses the sub-activities of “Functional Requirements Analysis” and “Navigational Requirements Analysis”. The former produces a *use case* model, and in the later the previous *use case* is complemented with the “navigation” capabilities

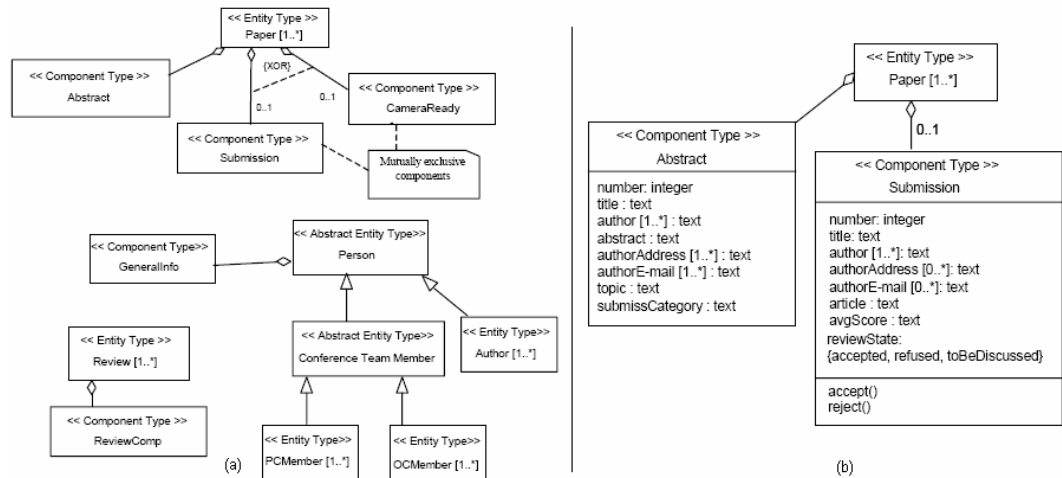


Figure 26: W2000's Hyperbase Information Design in-the-large(a) & in-the-small(b).

The "Hyperbase Navigation Design" defines the "navigational nodes" (nodes for short) and "navigational links" (links) of the application. Nodes, rendered using the UML stereotype node type, are derived from the structural design through a set of rules and design decisions. In the simplest case, nodes correspond to "leaf" classes. Figure 27 illustrates an example of the navigation from the "paper abstract" to the "paper submission" or to "paper camera" ready.

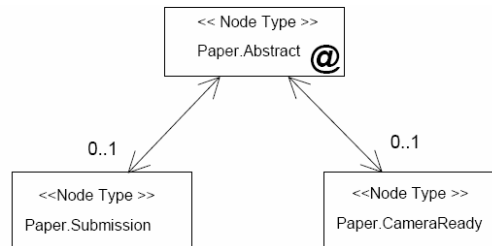


Figure 27: W2000's Hyperbase Navigation Design.

W2000 complements the *use cases model* with information relevant to the implementation of the business logic of the system which is an important implementation feature. The modeling of the information structure is made based on a class diagram. However, like most of the Hypermedia methodologies, the W2000's user interface navigation relies on a class diagram in a bottom-up approach that makes the navigation dependent from the information structure which in our opinion does not take advantage of the user-centered approach made by the modeling of *use cases*.

OMMMA

OMMMA [Stefan Sauer and Gregor Engels, 2001] is an object-oriented UML-based methodology for the development of IMDs which allows the modeling of domain (both information and media), navigation, system behavior (temporal and logical) and presentation. Briefly, this methodology covers all the design aspects of IMDs (modeling presentation, content and conceptual levels).

Figure 28 presents the class diagram that structures an educational application for organizing course material. The class diagram distinguishes the semantic part of the application and the media types deployed to present the content of Multimedia objects (that are marked using the stereotype <<media>>). The stereotype <<application>> is used to distinguish (Multimedia) application classes that correspond to Multimedia information from general application classes. The stereotype <<scenario>> marks classes of objects that represent complex scenarios, i.e., composite parts of the interactive Multimedia application that involve several <<application>> objects with temporal and spatial relationships.

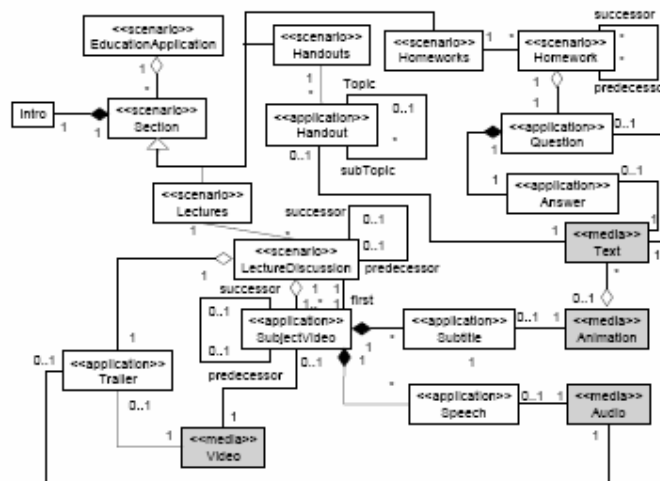


Figure 28: OMMMA’s class diagram for the modeling of scenarios, applications and media classes.

OMMMA models interaction (Figure 29) by means of a UML collaboration diagram. The user of the system is depicted by an *actor* that interacts with the modeled system. Users can only interact with objects of stereotypes <<interaction>> and <<presentation>> for input and output, respectively.

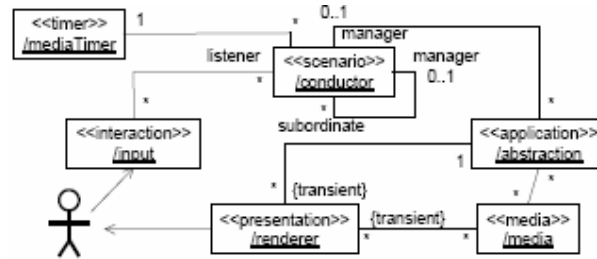


Figure 29: OMMMA's interaction modeling.

Furthermore, OMMMA separates reactive behavior from timed procedural behavior as two different modeling views of a Multimedia application. They are modeled by statechart diagrams (Figure 30) and by sequence diagrams (Figure 31) respectively.

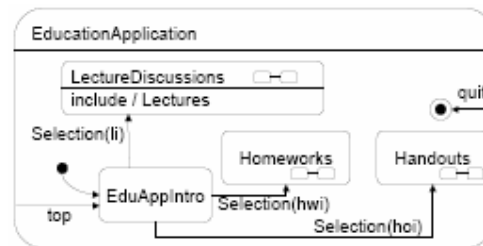


Figure 30: OMMMA's Statechart diagram modeling the top level reactive behavior of the education application.

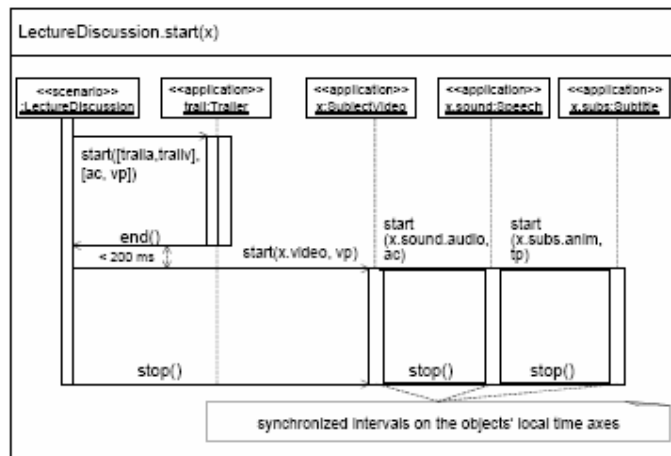


Figure 31: OMMMA's Sequence diagram modeling timed procedural behavior for the presentation of lecture discussion videos.

OMMMA models the basic IMD requirements, i.e., the presentation, the user interaction, the media synchronization and models the media content. However, OMMMA's solutions, especially the modeling of the presentation structure is very complex mixing the user interfaces with the media content resulting in a model with scalability problems.

Following the presentation of UWE, W2000 and OMMMA, we now evaluate the methodologies by the following criteria:

- Use Cases – if a *use case model* is produced or alternatively if user *task* analysis is made;
- Interface Design – (if the methodology includes) the modeling of the user interface;
- Navigation – modeling of the navigation among user interfaces;
- Interaction – modeling of the user interaction;
- System Responsibilities – modeling of the system behavior;
- Synchronization – modeling of the synchronization among media objects;
- Database– modeling of the information structure for purposes of database construction;
- Content– modeling of the media.

Table 2: Analysis and Design methodologies comparison.

	UWE	W2000	OMMMA
Use Cases	X	X	
Interface Design	X		X
Navigation	X	X	
Interaction	X	X	X
System Responsibilities		1	
Synchronization			X
Database	X	X	
Content			X

1 - Rules are defined in the *use cases* model for the access to the information that will be later implemented by the system responsibilities.

By the analysis of Table 2 we conclude that none of the contributions cover all the defined criteria, however, these methodologies complement themselves covering all the requirements. By further analysis of the table, it is possible to conclude that the Hypermedia derived methodologies (UWE and W2000) cover similarly the same requirements (W2000 does not provide interface design and UWE does not model *system responsibilities*). Complementarily, the requirements that are not covered by these approaches are covered by OMMMA that provides support for synchronization and content modeling.

II.6. CONCLUSIONS

This chapter presented the state of the art related to the existing modeling techniques for requirements identification, analysis and design of Interactive Information Systems with support for Multimedia.

From the techniques found in the literature for the definition of requirements, analysis and design of Interactive Information Systems, some served as basis for the contributions that will be presented in this thesis and some other may be used in the future evolution of these contributions.

The main contributions of this thesis are presented in the following Chapters III – Requirements (*Process Use Cases*) and IV – Analysis and Design (*MultiGoals*).

III. REQUIREMENTS (PROCESS USE CASES)

The identification of *use cases* is one key issue in the development of Interactive Information Systems. User participation in the development life cycle can be seen as critical to achieve usable systems and has proven its efficacy in the improvement of systems appropriateness. Indeed, the involvement of users in the requirements definition can add a significant improvement in both consecutive/interleaved tasks of: (i) understanding and specifying the context of use, and, (ii) specifying the user and organizational requirements as defined in Human-Centered Design (HCD) [International Standards Organization, 1999].

Existing solutions provide a way to identify *business processes* and/or *use cases* in order to achieve system definition, but they don't do it in an agile and structured way that helps to efficiently bridge Business Process Management and Software Engineering. *Process Use Cases* is a methodology, defined in the *Goals* software construction process, for the identification of *use cases* and information *entities* during the modeling and reorganization of *business processes* focusing the results in the identification of the functional requirements for the correct development of an Interactive Information System.

III.1.INTRODUCTION

In a competitive market, the ability of enterprises to make their services available to their clients and to be able to modify them easily might be an important advantage. Even in a small enterprise (e.g. 10 persons) *business processes* (BPs) can be complex including *tasks* in which performance, functionality and appropriateness (also called correctness of the software) can be crucial for success creating the need for system modifiability, most times with relevant time and cost constraints. In order to fully control the implemented services, the user *tasks* that support it and the software structure behind them, business processes (services), *use cases* (user *tasks*) and the architecture of the Interactive Information System (the software structure) must be documented.

The establishment of regular enterprise modeling activities for Business Processes Management (BPM) and Software Engineering (SE) enables bridging these two disciplines by means of a shared process (if the same notation is used). This connection happens where persons and system meet, the *use cases*.

In particular, the Unified Modeling Language (UML) [Object Management Group, 2003] provides a notation that encloses important concepts and diagrams that can be applied in both BPM and SE. Indeed, UML based techniques that make the mapping between BPs and Interactive Information Systems design using *use cases* already exist ([Jana Koehler et al., 2002], [Remco Dijkman and Stef Joosten, 2002]), however, in our opinion, not with the needed efficiency.

This chapter presents *Process Use Cases* (PUC), a methodology that guides the stakeholders of a software project from the initial idea until the identification of *use cases* during the identification and design (analysis, improvement, modeling and automation) of BPs.

This chapter is organized as follows: Section IV.2 introduces *Process Use Cases*. Section IV.3 presents the project used throughout section IV.4 to illustrate the methodology. Section IV.5 presents the conclusions.

III.2.PROCESS USE CASES: AN OVERVIEW

Process Use Cases (PUC) is a methodology defined within *Goals*, and is a solution to bind the phases of requirements identification and analysis rapidly, through the identification of *use cases* (functional requirements) and information *entities* as a leap to software analysis.

In order to achieve automation of the business processes, PUC covers partially the lifecycle of Business Process Management (Figure 32) [Webinterx, 2006] assuring that the BPs are analysed, improved and modeled before they are automated (Monitoring is out of the scope of PUC). The “analysis” is understood as the inspection of the current workflow of the BP, the “improvement” is the reorganization of the BP in a way that it becomes more efficient (for example by deciding which *tasks* to automate). After the “improvement”, the BP is “modeled” and finally it is “automated” by a Software Engineering process that leads to the development of an Interactive Information System.

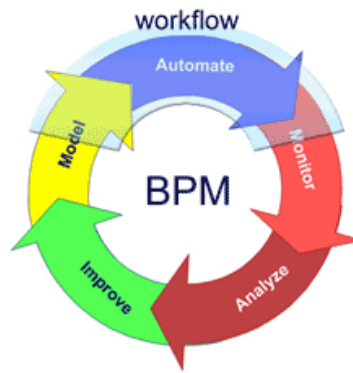


Figure 32: Business Process Management lifecycle.

PUC describes the development of 4 artifacts: 1 statement and 3 models (respectively *high-level concept*, *domain model*, *Business Process Model* and *process use cases model*) using an information-oriented strategy for the identification and association of the components generated: *business processes*, *information entities*, *actors* and *use cases*.

Consider Table 3 which enumerates the steps of PUC. Each step has a name (Interiorize Project, Information Identification, etc...) and produces one artifact (*high-level concept*, *domain model*, etc...) that is manipulated by an intervenient (architect, analyst and/or client) towards components definition (Entities, Business Processes, etc...).

Table 3: Steps of *Process Use Cases* methodology

Step	Step Name	Intervenant	Artifact Name	Components Manipulated
1	Interiorize Project	Architect, Client	High-Level Concept	N/A
2	Information Identification	Analyst, Client	Domain Model	Entities
3	Business Processes Identification	Analyst, Client	Business Process Model	Business Process, Entities, Actors
4	Use Case Identification	Architect, Analyst, Client	Process Use Cases Model	Tasks, Use Cases

To illustrate the main steps of this methodology, consider Figure 33 that depicts the business process that leads to the functional requirements identification which is the goal of PUC. Notice that the *domain model* and the *Business Process Model* are outputs of Step 2 and 3 respectively but can also serve as input for those phases meaning that these two phases can be iterative.

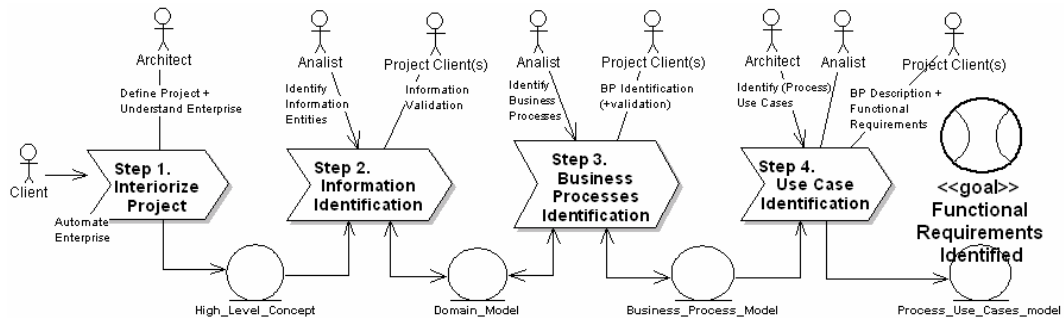


Figure 33: *Process Use Cases's* BP.

Goals suggests that a top-down, *use case*-driven, architectural centric analysis and/or design Software Engineering methodology follows the application of PUC, taking full advantage of the artifacts produced so far towards the construction of the Interactive Information System.

The following sections present a case study and its illustration throughout each step of *Process Use Cases*.

III.3. A PROJECT

In order to illustrate *Process Use Cases* (PUC), a project under development for a small enterprise is presented. This (non-profitable) enterprise related to a local governmental library (in Madeira, Portugal), is responsible for the bibliographic investigation on gastronomy (the project is referenced as “Gastronomy Project” along the thesis). The idea of the director is to divulgate the gastronomic events promoted by the enterprise and the existing gastronomic recipes in a website. However, the budget for the project is reduced and the development should be kept to its minimal.

After a first approach, in which an attempt was made to understand the main activities of the enterprise, it was possible to know which were the enterprises’ main products: the identification and cataloging of gastronomic recipes and the organization of gastronomic events.

By knowing the enterprises products we were able to produce the *High-Level Concept* contributing for the mutual agreement (with the client) on the mission of the project. After that, the entities identified in the *High-Level Concept* were combined in a *Domain Model* that later had the contribution of more information entities identified in the modeling of the *Business Process Model*. The *Business Process Model* identified 3 relevant business processes within the scope of the project and for each one was identified the inputs, outputs and the *actors* involved. Finally the 3 business processes were detailed with the *process use cases model* and the activities that needed automation were identified and transformed into *use cases* (the functional requirements for the Interactive Information System).

These steps of *Process Use Cases* are presented in the next sections.

III.4. THE STEPS OF PROCESS USE CASES

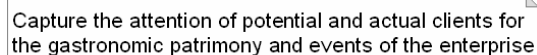
This section presents and illustrates each step of *Process Use Cases* using the project presented in the previous section.

III.4.1. Step 1 – Interiorize Project

The interiorize project is the only unstructured part of PUC. The *high-level concept* (HLC) is a paragraph (technology independent) that describes the part of the system (or full system) that is going to be implemented. The *high-level concept* must be understood by all the stakeholders (the community) of the project promoting a shared vision that will help the project community to keep focused on the product development.

In this step *client* and *architect* agree on a *high-level concept* for the project. To do this, it is important to understand the scope of the project within the enterprise global activity, so, it is necessary to understand how the enterprises' activities lead to the production of its main product(s) and what is the strategic reason that leads to the need of automation. Access to artifacts such as enterprise hierarchical organizational structure and legislation may provide important information, and by interviewing the clients' project manager, member preferably related to the enterprise's process of decision, sufficient information may already be compiled to produce the *high-level concept*.

In the project presented in this thesis the *high-level concept* agreed with the client (Figure 34) was: "Capture the attention of potential and actual clients for the gastronomic patrimony and events of the enterprise". The HLC expressed the intention of the enterprise to enlarge the number of clients and promote client fidelity by providing a quality service of information that combined the traditional historical recipes and the events that promoted those recipes.



Capture the attention of potential and actual clients for the gastronomic patrimony and events of the enterprise

Figure 34: Step 1- *High-level concept* for the project.

III.4.2. Step 2 - Information Identification

The information identification is a crucial step in the goal of achieving requirements definition. Based on the identified entities of information, it will be possible to identify the business processes that need to be modeled according to the objective of enterprise automation. At the same time, a *domain model* with the enterprises' entities of information that can be used in a Software Engineering process is already being modeled.

Information is very stable within an enterprise. Mainly, information manipulated by core *business processes* is persistent from the birth of the enterprise until its closure and is independent from the technology used to manipulate it. Information parts relate to each other naturally, and the objective is to produce a model, the *domain model*, that contains and relates all the identified parts.

In this step, the *analyst* identifies the main concepts of information defined in the *high-level concept*. These information concepts are transformed into *entities* that will be the first ones in the *domain model*. An *entity* is defined in *Wisdom* [Nuno Nunes, 2001] as a class used to model perdurable information (often persistent). It is also complemented that *entity* classes structure domain (or business) classes and associate behavior often representing a logical data structure. These *entities* represent information (not actions, actors, nor *business processes*; but they may coincide) and relate to each other by the composition of a meaningful structure. This structure has relations of hierarchy (inheritance), dependency (composition) and possession (association) and is called *class diagram* [Object Management Group, 2003].

The *domain model* is a classical class diagram as defined in UML [Object Management Group, 2003]. In PUC, the *entity* (which is a class) stereotype is used instead of the *class* stereotype which at this stage is a more accurate concept of information. This model (since it is described using a standard language, UML) can be used along all the Software Engineering process. At implementation stages, it is often used to generate database tables and (programmed) classes to manipulate these entities. The *domain model* must be updated at any stage in the process when new entities are revealed (particularly as a result of Step 3 in an iterative process).

The *domain model* is defined based on the information entities identified in the *high-level concept* statement. These information entities are placed in the *domain model* relating to each other according to the natural relation between information entities and their cardinality is also defined. After this first step, the *domain model* will be

updated whenever new information entities are identified during the modeling of the *Business Process Model* (Step 3).

It is suggested that the analyst describes the *class diagram* in natural language to the client in order to achieve diagram validation.

In the project presented in this thesis, the first *entities* derived from the *high-level concept* (Capture the attention of potential and actual clients for the gastronomic patrimony and events of the enterprise) were: “client”; “recipe” and “event”. The entity “client” existence, although implicitly related to the events, was reinforced when we noticed that the *business process* for recipe capture also involved donation of recipes by “clients”. The first *entities* identified were then combined with other *entities* (“Advertisement”, “Producers” and “Recipe Submitted for Approval”) identified in Step 3 (*Business Processes Identification*) to compose a single information structure as presented in Figure 35.

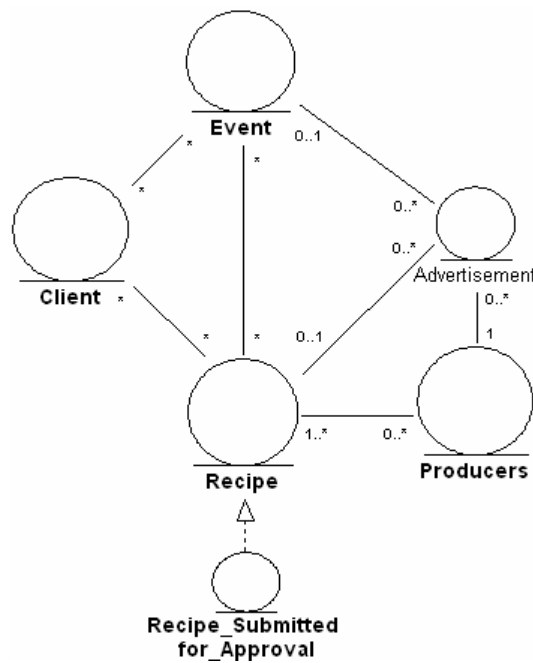


Figure 35: Step 2 - *Domain model* for the project.

III.4.3. Step 3 - Business Processes Identification

The objective of this step is to identify business processes for possible automation based on the information entities identified until this stage. At the same time, valuable information that can serve as documentation for future Business Process Management activities is being produced.

Business processes (BP) exist in an enterprise to achieve a certain objective, a goal, a product, that can be described by information (associated with this product). BPs happen as many times as exist the need to give response to the needs of some enterprise member or third party (e.g. client) with some responsibility (active or passive, with some relation to the enterprise) within the activity of the enterprise. Many enterprise members can interact with these processes by carrying out some complete, unitary *task*, in which many different *entities* can be manipulated (consumed or produced). In order to be able to control (e.g. reorganize) these BPs, it is important for an enterprise to maintain complete and detailed information of relations among BPs, their *inputs*, *outputs*, *actors* and triggering *events*.

In this step, *analyst* and *client* will identify, relate and detail *business processes*. The identification of BPs should take place, at least, from the business unit (in a hierarchical perspective) “directly” responsible for the information being managed, i.e. unit(s) that consume or produce this information to achieve complete and meaningful *tasks*. *Business processes* that relate “directly” to the information identified until this stage must be documented in order to provide the understanding of all the manipulation made over the identified information, if within the scope of the project defined in the *high-level concept*.

BPs are named according to their *goal* (the *product* of the BP), whether it is a service, information or a material product (e.g. product “television”, BP name “build TV”). BPs *products* are represented by *entities*, the associated information.

The persons that interact with the *business process* are called *actors* which are users that interact with a system. In *process use cases*, *business processes* are the “system”, and the stereotype used is the UML’s “user”. *Actors* are associated with BPs using association and their objective(s) are written in natural language (e.g. “approve recipe”) separated by a plus signal (+) naming the association. When an *actor* triggers the *business process*, an *event* is generated and its relation with the *business process* is represented with a flow (arrow form).

The *outputs* and *inputs* (information, resource and output in the *Business Process Model* (Eriksson, 2001 #2)) are represented by *entities*. *Business processes* can be related to each other, i.e., the outcome of a business process (which is an *event*) serves as the income to the next one providing an information *entity* shared by the two BPs in a horizontal hierarchy. When the flow is towards the *business process* it is an *input* (and generates an *event*) and the contrary direction represents an *output*. Associations can be bi-directional representing *event*, *input* and *output* in both directions.

In the project presented in this chapter, 4 *business processes* that directly manipulated the *entities* “client”; “recipe” and “event” (Step 2) were identified: “Obtain Recipe”; “Make Event”; “Advertise” and “Obtain Gastronomic Information”.

The “Obtain Recipe” business process (Figure 36) generates the information for the entity “recipe”. In this business process a donator or a gastronomy investigator submit a new recipe (“recipe submitted for approval”) that is approved or not by a gastronomy consultant according to its authenticity.

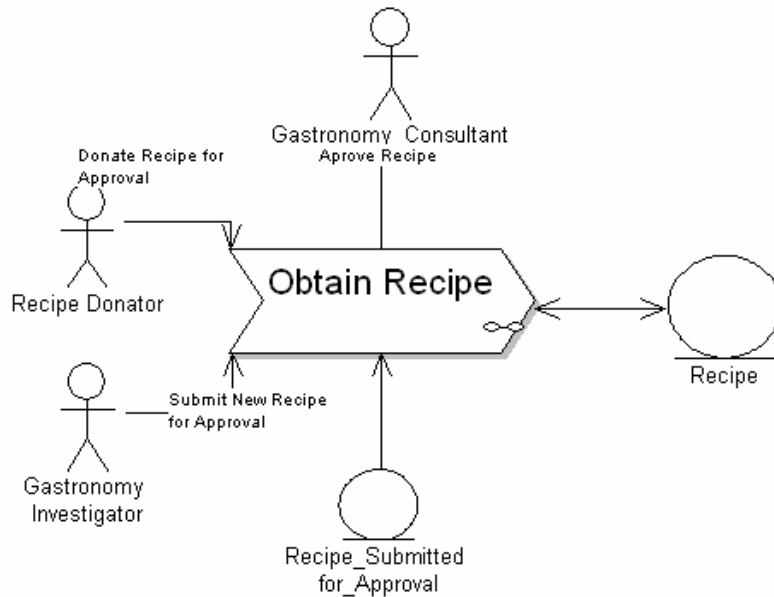


Figure 36: *Business Process Model* for the “Obtain Recipe” business process.

The “make event” business process (Figure 37) generates information for the entity “event”. In this BP the business manager and the event organizer interact to create a new event using the “producer” and “recipe” entities.

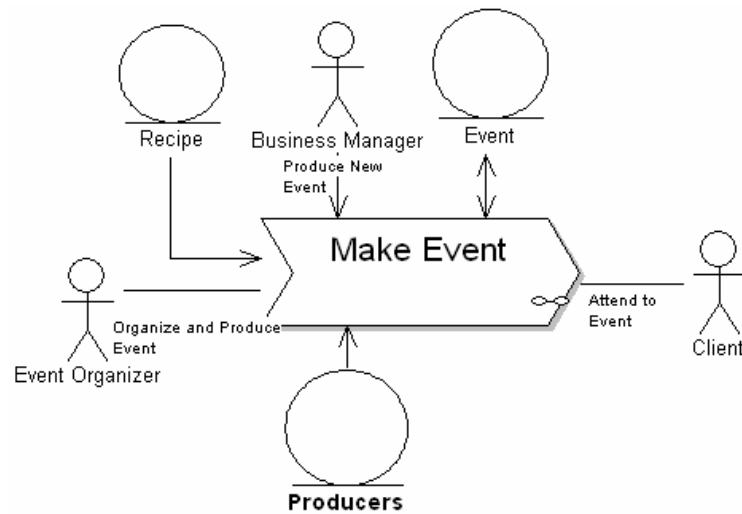


Figure 37: *Business Process Model* for the “Make Event” business process.

The “advertise” business process (Figure 38) was created in order to produce information for the website represented by the entity “advertisement”. In this BP the business manager delivers advertisements to the advertiser about recipes, events or generalized news.

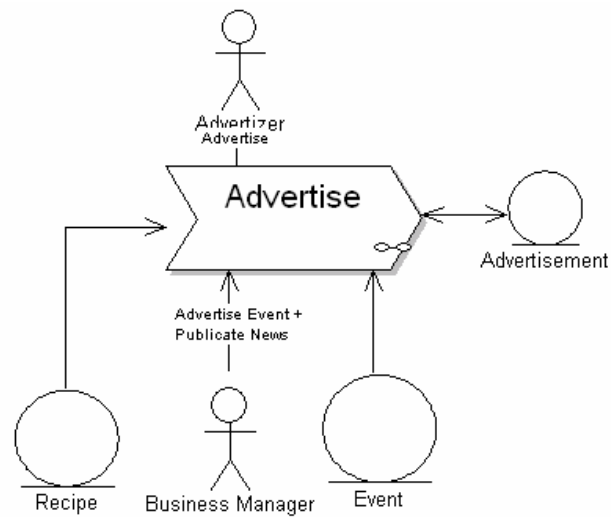


Figure 38: *Business Process Model* for the “Advertise” business process

The “obtain gastronomic information” is a new business process (Figure 39) that will exist as a consequence of the new website and that represents the usage of that website by the clients of the enterprise.

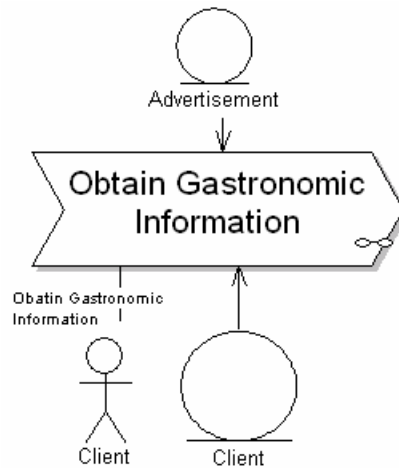


Figure 39: *Business Process Model* for the “Obtain Gastronomic Information” business process.

Figure 40 depicts the complete *Business Process Model* for the project. The business processes previously identified relate naturally to each other by sharing the entities of information and *actors* can relate to more than one BP with different objectives.

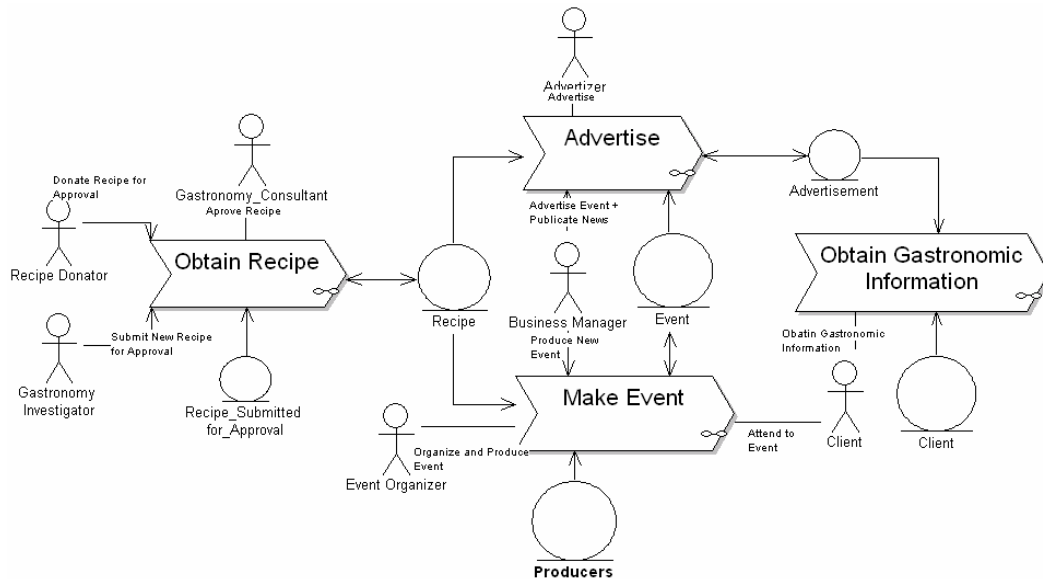


Figure 40: Step 3 – The *Business Process Model* for the project.

After the *Business Process Model* was designed the *client* validated the diagram and the *domain model* was updated. A new business process “certify producer” was identified based on the entity “producer”, however, since this BP was out of the scope of the project it was not documented.

III.4.4. Step 4 - Use Cases Identification

The identification of *use cases* is the purpose of this step. The business processes identified in the previous step will now be detailed using an *activity diagram* in which the activities that need automation will be transformed into *use cases* providing the projects' functional requirements.

The documentation of *business processes* in a language that every intervenient (stakeholders of a project) understands is important to enable correct dialogue over the *actors*, activities (*tasks*) and *goals* of the BP. BPs can be partially or completely automated or not automated at all.

In this step, *analyst* and *client* model the *tasks* (activities) of the *business process* which will be performed by the *actors* along the BP until achieving the targeted *goal*. A *task* (*task case*, as defined in *Usage-centered design* [Larry Constantine, 2006]) represents a single, discrete user intention in interaction with a system that is complete and meaningful. For instance, an *essential use case* which is defined by the same author as a specially structured form of a *use case*, called *essential (use case)*, that is, abstract, simplified, and independent of assumptions about technology or implementation.

The BP is designed with the *process use cases model*, through the use of an UMLs' *activity diagram* [Object Management Group, 2003] with swimlanes. The *tasks* the *actors* carry out are placed in the same swimlane. The *activity diagram* begins with an "initial" stereotype and ends with a "final" stereotype. The *transition* relation is used between *tasks*. UML's *activity* stereotype is used to represent *tasks* of the BP which are not *use cases*. *Fork* and *decision* are used to represent parallel activities and decision points, respectively.

Once all *activities* are identified, it is important that the *architect* (with the *client*) decides which *tasks* should be automated. When this happens, a *use case* (stereotype change) takes the place of that *activity*.

In the project presented in this thesis, based on the analysis of the models produced until the previous step (Step 3), we noticed (with the cooperation of the client) that the BPs mostly able to contribute with relevant information for the website were "Obtain Recipes" and "Advertise". In another perspective, "Obtain Recipes" could provide more valuable information for the website than "Make Event", and by means of generalization of the *task* "Advertise", support could also be achieved to advertise "news" about "recipes" and "events".

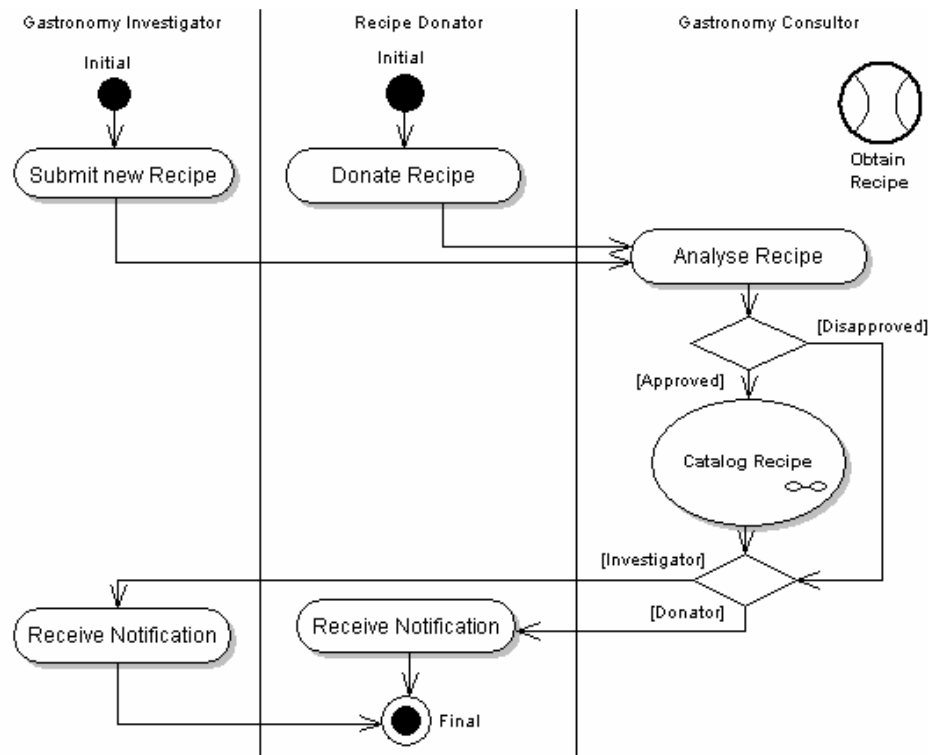


Figure 41: Process use cases model for “Obtain Recipe” business process.

Following the analysis of the *Business Process Model* (Step 3) the business processes were designed according to the *process use cases model*. Figure 41 depicts the design of the business process “Obtain Recipe”, where in this business process the activity “Catalog Recipe” was transformed into an *use case* (automated) for the purpose of obtaining recipes information for the IIS.

Figure 42 depicts the business process “Make Event” in which the activity “Organize Event” was eligible for automation. However, complex development was needed resulting in more man/hour than what could be supported by the existing budget. This was concluded once the management of the hired producers should be made in this *task*, and for this reason it was decided that the information about events would be published by means of an advertisement.

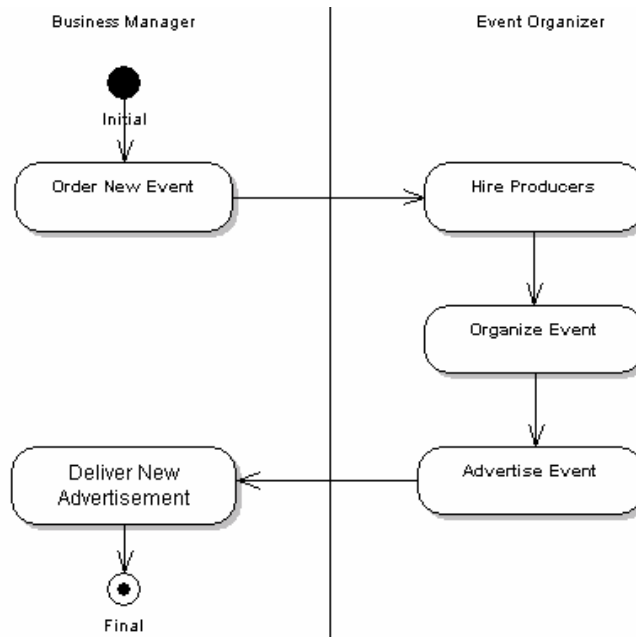


Figure 42: *Process use cases model* for “Make Event” business process.

Figure 43 depicts the business process “Advertise” which was created in order to provide information for the website. It was defined that the automated activity “Advertise” should be able to support information from Events, Recipes and general Advertisements.

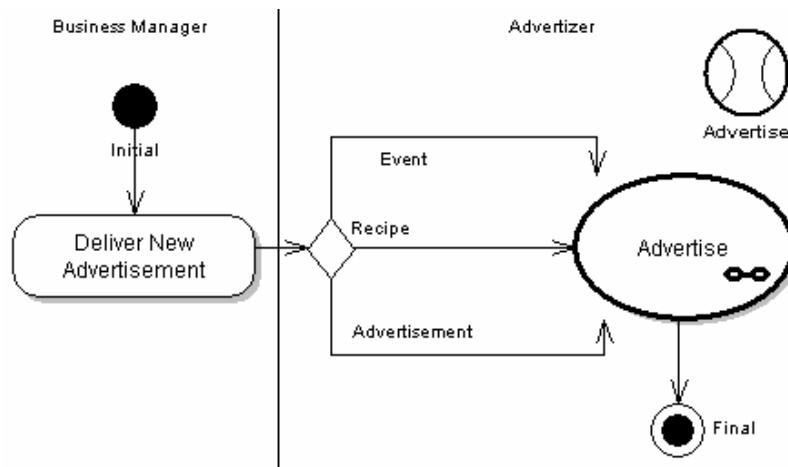


Figure 43: *Process use cases model* for “Advertise” business process.

Figure 44 depicts the “Obtain Gastronomic Information” business process. This business process represents the website manipulation made by a user (client or not of the enterprise) when searching for the gastronomic information provided by the website.

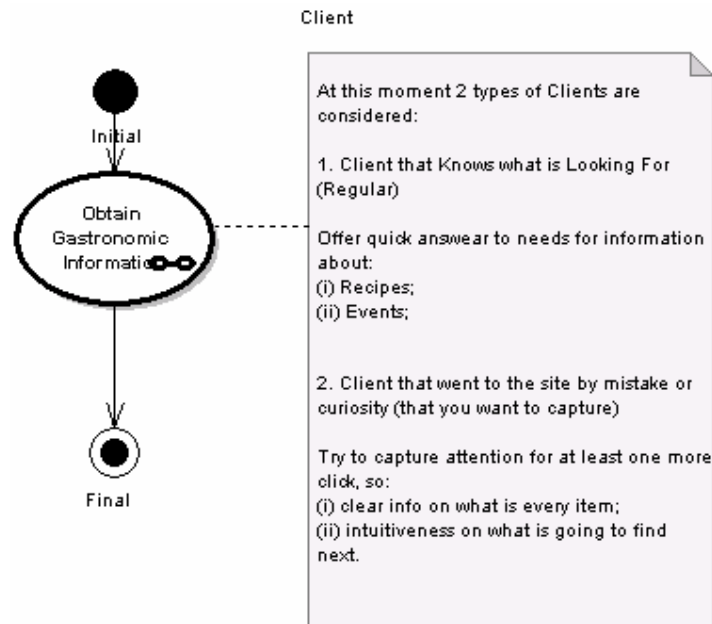


Figure 44: *Process use cases model* for “Obtain Gastronomic Information” business process

Process use cases is the model where users and Interactive Information System meet. However, it is not the purpose of PUC to establish the relation between *use cases* and entities. This is a task left for a Software Engineering process which carries along the information generated until this stage and brings consistency to this relation in later stages of that process.

III.5. CONCLUSIONS

Process Use Cases (PUC) is a methodology that identifies *use cases* as a leap for software construction producing valid artifacts for both activities of Business Process Management and Software Engineering. PUC has been already applied in over 10 different real software development projects for the Information and Computing Center in University of Madeira (UMa), Portugal, for the automation of at least one *business process* per project. It has been applied by both undergraduate students and IT professionals and shared with UMa managers for both Business Process Management and Software Engineering activities always resulting in a firm artifact that promoted consensus between the stakeholders.

In a modeling perspective, to achieve the more appropriate level of abstraction naming the *use cases* can be a very difficult task in Software Engineering if no global comprehension exists of the scope of the project within the enterprise organization. Using PUC it is easier to reach the appropriate abstraction to nominate the (*essential*) *use cases* in a way that they make sense in both Business Process Management and Software Engineering disciplines. This is possible through the definition of compatible formalizations of the stereotypes used (*entities, users, business processes, activities* and *use cases*), that are provided by LUCID [Charles Kreitzberg, 1999], *Wisdom* [Nuno Nunes, 2001] and *Usage-centered design* [Larry Constantine, 2006], producing a notation also suitable for the application of agile software analysis and design methods.

IV. ANALYSIS & DESIGN (MULTIGOALS)

The development of Interactive Information Systems has largely benefited from the improvements made in the field of Human-Computer Interaction (HCI) as a way to produce usable enterprise systems that solve with relative success the needs for automation. The internet has increased the possibilities of communication beyond the limits of the enterprises' local network representing a breakthrough for the product towards the potential and actual clients. This phenomenon has inspired the appearance of attractive technology to enrich web pages (that present the products) in which Multimedia stands in a relevant place.

However, this recent crescent complexity of enterprise Interactive Information Systems does not have a correspondence in the existing Software Engineering, Multimedia or Hypermedia methods so that this integrated complexity can be controlled and designed.

MultiGoals is a methodology for the analysis and design of complex Interactive Information Systems (IIS) that describes the components of the application in detail: user interface; system behavior and content, and that defines the usage of patterns for the combination of both Interactive Information Systems applications and interactive Multimedia documents.

IV.1. INTRODUCTION

This chapter presents *MultiGoals*, a methodology to guide the authoring of complex applications for both Interactive Information Systems (IISs) and Interactive Multimedia Documents (IISs). The result of the *MultiGoals* will yet be an hybrid IIS application but with support for Multimedia attractive capability to manipulate synchronized user interfaces media with or without user interaction.

MultiGoals defends the usage of hybrid application logic patterns in a way that they can combine themselves to produce dynamic system behavior from the combination of hybrid Interactive Information Systems' business logic and interactive Multimedia documents' behavior. All the identified components are grouped into a single and complete application structure assuring the traceability from *use cases* to code generation.

This chapter is organized as follows: Section IV.3 is an overview of *MultiGoals*, Section IV.4 presents an example application used throughout Section IV.5 to present the Steps of *MultiGoals* in detail. Section IV.6 presents the conclusions. The stereotypes used in *MultiGoals* are presented in Appendix B.

IV.2. MULTIGOALS: AN OVERVIEW

MultiGoals is a methodology for the modeling of applications with support for Multimedia. The result of the application of *MultiGoals* can be: (i) an Interactive Information System (IIS), i.e., a traditional Software Engineering application; (ii) a Multimedia Application (Multimedia player, video-conference, etc...); (iii) an Interactive Multimedia Document (IIS); or (iv) an Hybrid Application with both support for IIS records manipulation and Multimedia presentation.

Table 4: Steps of *MultiGoals* methodology

Step	Model Name	Components Manipulated	Phase	IIS Design Level
1 *	Use Case	Actor, Use Case	Analysis	Requirements
2 *	Activity Diagram	Interaction Space, Task, System Responsibility	Analysis	Requirements
3 *	Interaction Model	Task, System Responsibility	Analysis	User Interaction
4	Navigational Model	Interaction Space	Design	Presentation
5 *	Presentation Model	Interaction Space, Task	Design	Presentation, User Interaction
6	Application Domain Model	Entity	Design	Presentation, Content
7	Application Object Model	Entity Object	Detailed Design	Presentation, Content
8	Conceptual Model	Interaction Space, Task, System Responsibility, Entity, Entity Object	Detailed Design	Conceptual, Content
9	System Behavior Model	System Responsibility	Detailed Design (Multimedia)	Conceptual
10	Temporal Model	Task, System Responsibility	Detailed Design (Multimedia)	Conceptual
11	Multimedia Architecture	Interaction Space, Task, System Responsibility, Entity, Entity Object	Detailed Design	Conceptual

* *MultiGoals* simplified version.

As described in table 4 *MultiGoals* is composed of 11 steps. Each step adopts a different modeling technique (*use case*, *activity diagram*, *interaction model*, etc...) to produce the appropriate component (*actor*, *task*, *system responsibility*, etc...) that will lead to the design of the application. Indeed, these steps can be followed differently according to the level of detail needed. During the phase of analysis the author works

on the comprehension of the problem to be solved based on the *tasks* the user carries out to accomplish his objective; during the phase of design the author will start the conception of the system that will support the user *tasks* that solve the problem, and; during the phase of detailed design the author will detail each component of the system identified until that moment. The phase of detailed design is complementary to the phase of design, for this reason it is not mandatory to achieve acceptable system definition regarding system development. Note that models 9 (*system behavior model*) and 10 (*temporal model*) exist only for describing Multimedia documents.

The simplified version of *MultiGoals* is conceived as a fast solution to model the most essential issues of an IIS. Based on the analysis of the user *tasks* (*use cases*, *activity diagrams* and *interaction model*) the author will be able to model the user interface and identify relevant *system responsibilities*.

The *MultiGoals* simplified version is composed by the following models: (i) Model 1. - *Use cases model*; (ii) Model 2 - *Activity diagrams*; (iii) Model 3 - *Interaction model*, and; (iv) Model 5 - *Presentation model*. This version is directed to authors with few or no experience on Software Engineering methods and because of that the modeling of the domain was left out of the simplified version. It is intended to be used when there are relevant time constraints and when the problem to be solved is relatively simple, i.e., when complex system behavior is not expected, when Multimedia requirements are kept to a minimum (two media, one dynamic and one static) or when *system responsibilities* are simple (select and set data directly to database tables).

The application of this simplified version of *MultiGoals* will be presented in a case study in the next chapter.

IV.3. AN APPLICATION

The methodology based on UML presented in this chapter can be applied to support the design of complex applications, however, in order to illustrate its application, a simple, although real project application is used and presented in Figure 45.

The application used as example is part of an undergoing real project for an enterprise in the gastronomy business (presented in the previous chapter) in which the objective is the construction of an Interactive Information System that internally collects information relative to recipes and events and displays it in a website. In this chapter we illustrate *MultiGoals* by the application of the *use case* “Catalog Recipe” which was identified in the “Obtain Recipe” business process as a result of the application of the *Process Use Cases* methodology and defines the edition of a previously chosen “recipe” in which the “name”, “type” (category of the recipe), “ingredients” (extensive list of ingredients) and “directions” (the preparation of the recipe) must be defined. As a complement, a video can also be included in the presentation. As additional requirements for this example and as a way to illustrate the methodology with more completeness, the text of the area dedicated to the recipe type must be made with background color #FF6633 (Orange) and an audio sequence “ping-splash” must be played to give feedback to the user of the availability of the application.

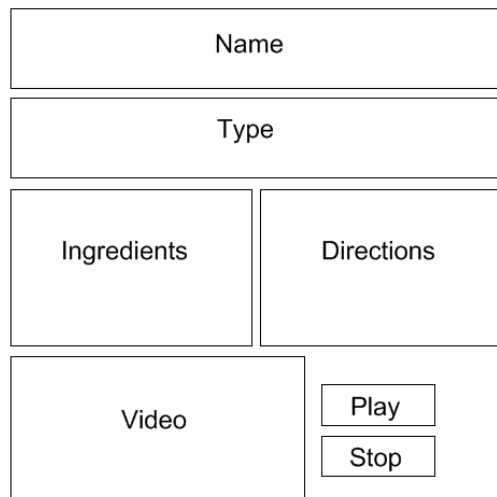


Figure 45: Illustration of an interactive Multimedia scenario

The main issues related to the design of applications using *MultiGoals* are addressed in the sequel. The application of *MultiGoals* to the remaining *use cases* identified previously is illustrated in the next chapter.

IV.4. THE STEPS OF MULTI GOALS

This section describes and illustrates each step of *MultiGoals* using the example given in the previous section.

IV.4.1. Step 1 – Use Cases

An (*essential*) *use case* identifies a part of the application that will solve some specific problem, and represents a single, discrete, complete, meaningful, and well-defined *task* of interest to an external user as defined by Larry Constantine in [Larry Constantine and Lucy Lockwood, 2000]. The *use cases model* in *MultiGoals* follows the classical semantics and notation for the UMLs' *use case diagram* [Object Management Group, 2003].

In order to specify the *use case*, it is necessary to identify the *user(s)*. The *user* represents a single person or a group of persons that want to achieve a goal (*task*), and more than one group of *users* can be related to the same *use case*. Notice that *use cases* can relate to one another (extends - when one *use case* complements another, or include - when a *use case* needs to include another one).

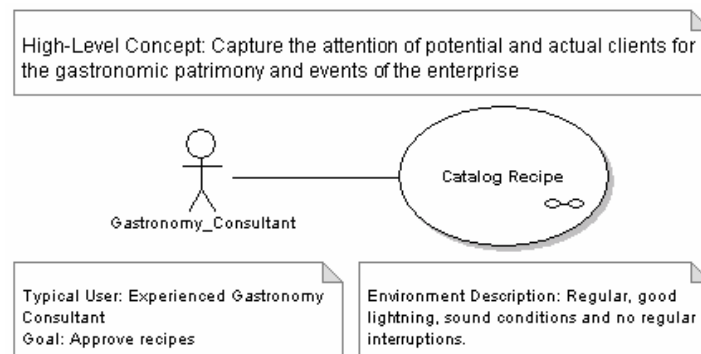


Figure 46: Step 1: *Use Case* and complementary information for the example application.

In order to complete the understanding of the usage, the *use case* should be complemented with the *High Level Concept* (defined in the first step of *Process Use Cases* – Chapter III) which defines in one sentence what the complete application should do (not only this *use case*). It is a statement defined in the LUCID Framework (Logical User Centered Interaction Design) [Charles Kreitzberg, 1999] as the first step for the envisioning of a product. The *high-level concept* is seen as a mission statement for a product to help focus on the product development.

The *high-level concept* for the example is related to the existence of a gastronomic patrimony and realization of events as a way to capture clients. The *use case* deals with the gastronomic patrimony which is maintained by a gastronomy consultant who has the objective of compiling the recipes of the enterprise (cataloging). The typical user of this *use case* will be an experienced gastronomy consultant whose objective is to catalog recipes.

IV.4.2. Step 2 – Activity Diagram (Interaction Spaces + Tasks)

The *activity diagram* will specify how the interaction between the user and the system will happen, what is supposed to happen in each side (user and system) that will lead to the accomplishment of the *task*. The *Activity diagram* follows the definition of the decomposition of an *essential use case* into “user intentions and *system responsibilities* in the course of accomplishing that *task (use case)*, described in abstract, technology-free, implementation independent terms using the language of the application domain and of external users” as defined by Larry Constantine in [Larry Constantine and Lucy Lockwood, 2000]. Since *MultiGoals* is tailored for the modeling of detailed interfaces, the name of the *interaction spaces* where the user actions (*tasks*) occur should be identified. The *activity diagram* for *MultiGoals* follows the classical semantics defined in UML’s *activity diagram* [Object Management Group, 2003], but in contrast, the classical notation of activities and sub-activities is replaced by the stereotypes of *task* (user intention) and control (*system responsibility*). The addition of the *interaction space* stereotype is a complement to the *activity diagram*.

The *activity diagram* is separated into *user intentions* and *system responsibilities*. *User intentions* are *tasks* that the *user* wants to accomplish on the system, which at this stage can be or not by means of direct user interactions, being most of the times a high level *task* representing what the user is doing at this step in order to complete his *task* (e.g. “Reserve Room”). Contrarily, *system responsibilities* are the response of the system to the *task* carried out by the *user* (e.g. “Confirm Room Reservation”).

The *activity diagram* can begin in either side, system or user. Usually, in common cases, 2 to 6 *tasks* are enough so that *user* is able to accomplish what he needs. Of course, the number of *tasks* depends on the complexity of the overall *use case* purpose. In general, more than one *user task* and more than one *system responsibility* can be executed in sequence.

After the *activity diagram* is completed with *tasks* and *system responsibilities* the *interaction spaces* (user interfaces) in which the *tasks* will be performed must be identified. Notice that one *interaction space* can support one or more *tasks*.

For instance, consider the *activity diagram* associated with the previous application which is depicted in Figure 47. In this diagram, the *user intentions* initially describe the intention of the user to “catalog a recipe” which is expressed in the “menu” *interaction space* and immediately carried out by the system returning the recipes in the “Recipe Browser” *interaction space* so that the user can select the recipe to edit or select a new

recipe. After selecting the recipe, the system will return the recipe cataloging tool ("Recipe" *interaction space*) to the user where the edition of the recipe will be made.

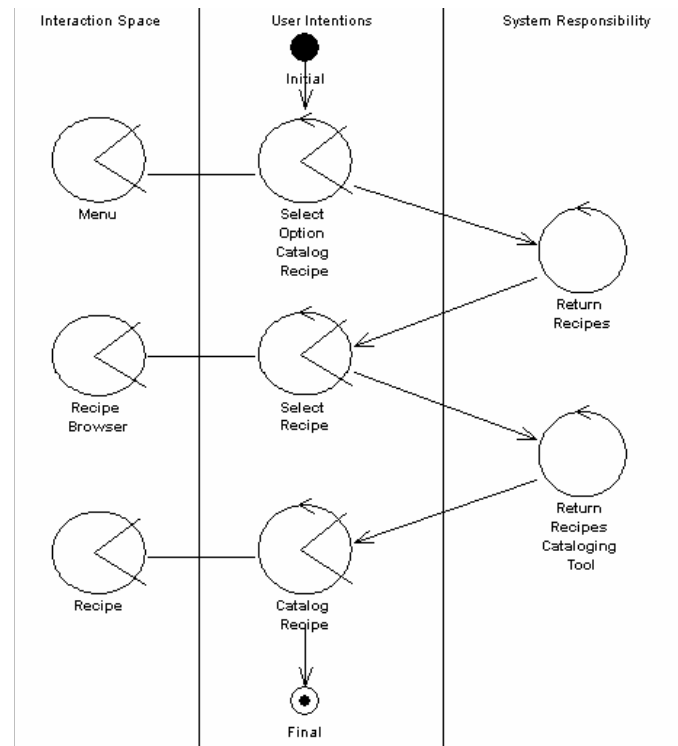


Figure 47: Step 2 - Activity diagram for the example application.

IV.4.3. Step 3 – Interaction Model (Task + System Responsibility)

The *interaction model* details the user interaction in order to perform a *task* (identified in the previous step) and specifies which will be the response of the system to each one of the user (sub) *tasks*, relating these *tasks* to the *interaction spaces* where they occur.

The *interaction model* details (decomposes) *tasks* into *sub-tasks*, and the corresponding *system responsibilities* into *sub-system responsibilities*. The higher level of an *interaction model diagram* is a combination *task* -> *system responsibility* taken from the *activity diagram* presented in step 2. Thus, a *task* from the *activity diagram* is decomposed by means of *concur task trees* (CTT) [Fábio Paternò et al., 1997] up to the description of an interaction on the *user interface*. Similarly, corresponding *system responsibilities* (which are controls, system functions) are decomposed (if needed) into lower level controls, which are executed whenever that *user task* takes place. The *sub-user tasks* are then associated with the corresponding *sub-system responsibilities*.

The decomposition of *tasks* into *sub-tasks* is carried out using aggregation, e.g. “Reserve Room” decomposes into “Select Room”, “Select Customer” and “Select Duration”. Moreover, an operator also must be specified among the *sub-tasks* in order to determine their order. These operators can be [Fábio Paternò et al., 1997]: *Choice* (T1 [] T2); *Independent concurrency* (T1 || T2); *Disabling* (T1 [> T2); *Enabling* (T1 > T2); *Suspend/Resume* (T1 |> T2); *Order independent* (T1 |= T2). For further information on these operators see [Fábio Paternò et al., 1997].

In the *interaction model*, the *system responsibilities* are the response of the system to a user interaction, thus, specific system response can be described at this stage. For that purpose, the *system responsibilities* can be defined according to the Static Patterns defined in Appendix A.

This decomposition should be made until reaching specific interaction with the system (e.g. “confirm reservation”, which means clicking a button). Then, such as in the *activity diagram*, it is necessary to identify in which *user interfaces* (*sub-interaction spaces* in this diagram) the *tasks* will be performed.

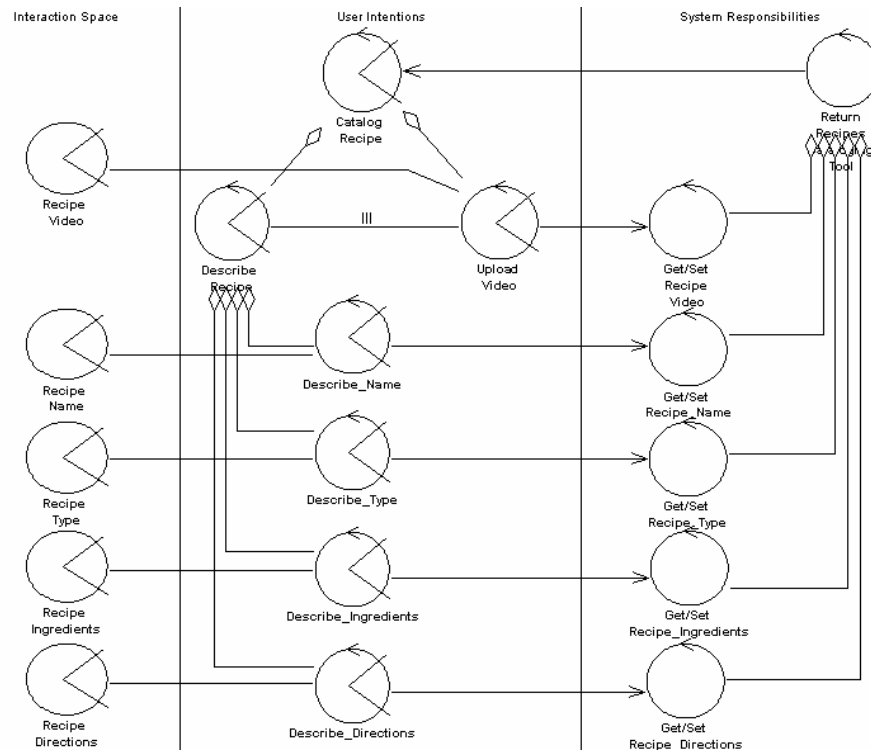


Figure 48: Step 3 - *Interaction model* for the example application.

For instance, Figure 48 illustrates the *interaction model* for the previous application. According to this figure, the combination “Catalog Recipe”-“Return Recipe Cataloging Tool” is placed at top of the diagram and all the *tasks* are decomposed until reaching user interaction. These user interactions (which are user intentions) are then associated with the corresponding *system responsibility* that will produce the necessary system response to the user intention. After this, the user interactions (user intentions) are associated with the *interaction spaces* where they take place.

IV.4.4. Step 4 – Navigational Model (Interaction Spaces)

The *navigational model* combines in one model all the *interaction spaces* identified in the previous diagrams and relates them into a single structure specifying the different possibilities of navigation.

In this model, two kinds of relations are used among the existing *interaction spaces*: Navigate and Contains. These are the UML extensions used in the *navigational model* which are specified in *Wisdoms' Presentation Model* [Nuno Nunes, 2001]:

<<Navigate>> is an association stereotype between two interaction classes denoting a user moving from one *interaction space* to another. The navigate association can be unidirectional or bi-directional (which means there is an implied return in the navigation).

<<Contains>> is an association stereotype between two *interaction space* classes denoting that the source class (container) contains the target class (content). The "contains" association can only be used between *interaction space* classes and is unidirectional.

The construction of the *navigational model* starts with the *interaction spaces* identified in Step 2 – *Activity Diagram*. If more than one *interaction space* is identified in Step 2 then those *interaction spaces* can be aggregated into a higher-level *interaction space* which will name the application built for the current *use case*. After that, the *interaction spaces* identified in Step 2 are decomposed into the *interaction spaces* originally identified in Step 3. The navigation between *interaction spaces* (of the same level or not) will occur when an *interaction space* replaces another.

For instance, in the application presented in this chapter, the *interaction spaces* "Menu", "Recipe Browser" and "Recipe" taken from Step 2 – *Activity Diagram*, were aggregated into a higher-level *interaction space* called "Gastronomy Application". The *interaction spaces* identified in Step 3 – *Interaction Model* were then aggregated into the corresponding *interaction space* "Recipe". This is illustrated in Figure 49.

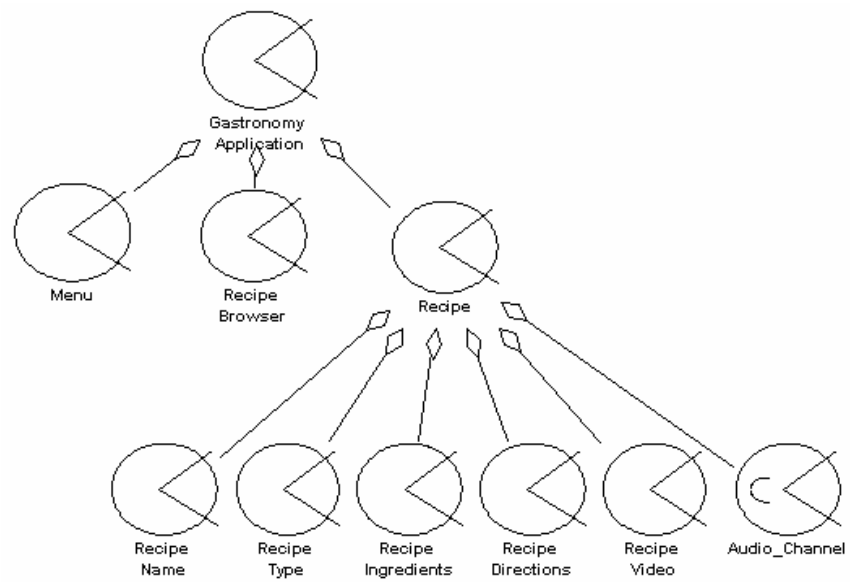


Figure 49: Step 4 - *Navigational model* for the example application.

IV.4.5. Step 5 – Presentation Model (Interaction Spaces + Tasks)

The *presentation model* is the design of the *interaction spaces* of the application in terms of the aspect (spatial layout) of each *interaction space* and its associated functionality.

The *presentation model* is based on the *interaction spaces* identified in the *navigational model* (Step 4), and the visual aspect of each area is modeled using *Canonical Abstract Prototypes* (CAPs) [Larry Constantine, 2003]. The functionalities, which are the user *tasks* supported by the application, are associated with the visual representation of each *interaction space*. The *audio interaction spaces* are represented by the class stereotype of the *interaction spaces*.

CAPs allow the modeling of a complete set of user interactions that can occur in the components of the user interface. For instance, for the selection of a room, a list must be displayed and its appropriate CAP used (abstract selectable collection material, should be used to specify this situation). CAPs are applied to detail all the *interaction spaces* of an Interactive Information System.

The *interaction spaces* identified are represented by *regions* (which describe spatial coordinates for the presentation of *interaction spaces*) and are identified with the *interaction space* name within parenthesis. The user *tasks* performed in each *interaction space* are placed inside the area of the *interaction space* if possible; if not, they are associated with the *interaction space* with a dashed line. *Audio interaction spaces* are represented by a class with the *audio task* stereotype.

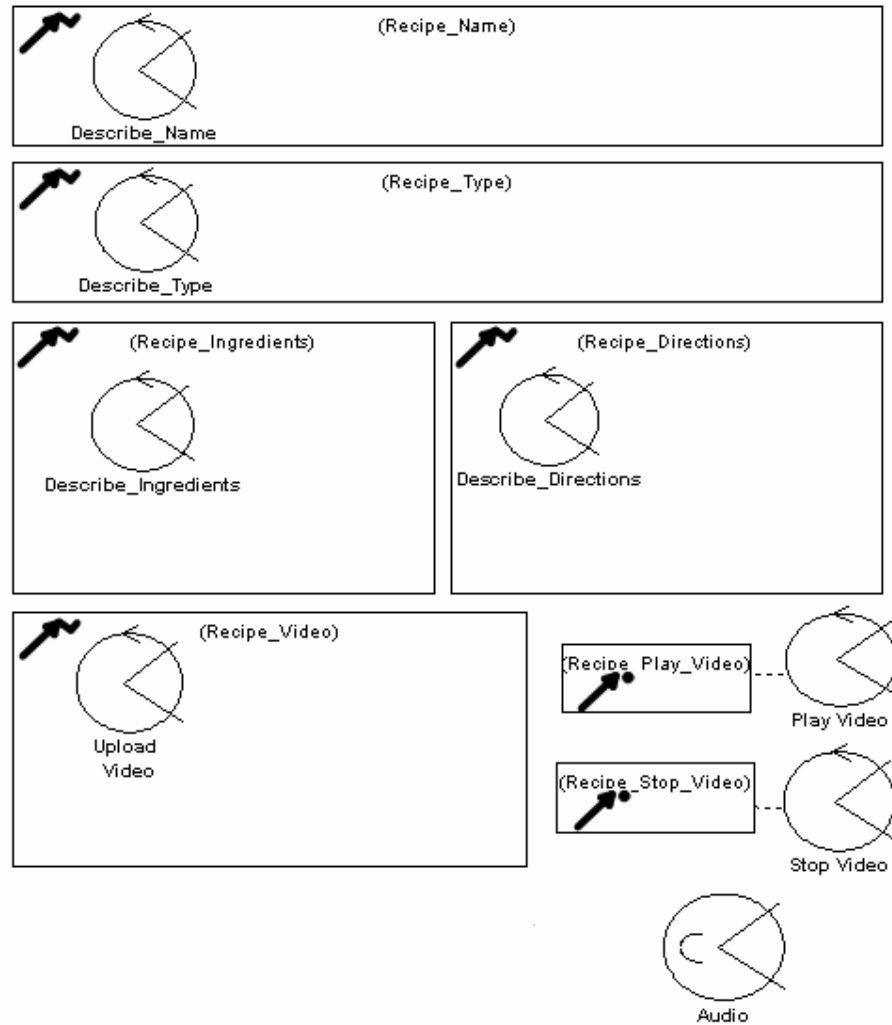


Figure 50: Step 5 – *Presentation model* for the example application.

For instance, consider Figure 50 which describes the *presentation model* for the previous Application. In the description of this picture “Recipe Name”; “Recipe Type”; “Recipe Ingredients”; “Recipe Directions”; “Recipe Video” are *interaction spaces* where a modify action will take place (CAP). Note that in “Recipe Video” this is also a modify action where the modification will be made over a media instead of text, and that the *interaction spaces* “Recipe Play Video” and “Recipe Stop Video” will have a direct action over the “Recipe Video” media. The audio *interaction space* “Audio” will present the sequence “Ping-Splash”.

IV.4.6. Step 6 – Application Domain Model (Entities)

The *application domain model* describes the persistent structure for the content of the application in its perspectives of user interface *interaction space* values, user interface media values and user information values. It describes the information concerning each visual region (*regions domain model*), the content of each media object composing the application (*media domain model*), and the semantic composition of the application (*ontological domain model*).

The *application domain model* is described through three phases:

- The *regions domain model* specifies the attributes of each *region*. Thus, the *interaction spaces* identified in the *presentation model* (Step 5) are transformed into generic UML classes (alternatively, the representation can also be made using the *interaction space* stereotypes). For this purpose, the necessary attributes for each *region/interaction space* were defined, and a pattern was proposed for each kind of *region* (visual or audio *interaction space*) with their appropriate attributes.
- The *media domain model* defines which media will be presented in each *interaction space* of the application. For this purpose, a pattern was proposed for each kind of media (audio, video, image, text, etc...). Each class of the *media domain model* is associated with the corresponding region and the cardinality is defined.
- The *ontological domain model* relates to the concrete things (entities) and conceptual things (concepts) that support the user information that will be presented by the application. In the *ontological domain model* (can be a *domain model* that has already been drawn in a previous phase of analysis or requirements) the classes must have their attributes defined and relations among classes must have their cardinality and optionally their names (of the relations) defined.

It is important to emphasize that in order to illustrate all the Multimedia features for the design of the application, the attributes for the definition of the *regions domain model* and *media domain model* were based on the description of these components from the language SMIL [The World Wide Web Consortium, 2007].

The description of the *application domain model* begins with the *regions domain model*. In this part of the model the *interaction spaces* detailed in Step 5 - Presentation Model are now represented by classes (with attributes) that correspond to “regions” that follow the composition defined in Step 4 - Navigational Model. The next step is the *media domain model*, in this part of the model each “region” is associated with a class

representing the type of media related to it and the cardinality of that association. The media attributes defined for the *media domain model* classes also follow the SMIL convention for the definition of the class attributes. The last step is the *ontological domain model*, this diagram follows the classical UMLs' class diagram [Object Management Group, 2003] that relates the concrete and conceptual things manipulated by the application. These classes are then associated with the *media domain model* classes and their cardinality are defined according to needs of the user interface in terms of media for each *ontological domain model* Class, i.e. for each part of the user interface that will contain a media with information from the *ontological domain model* an association is made and its cardinality is defined.

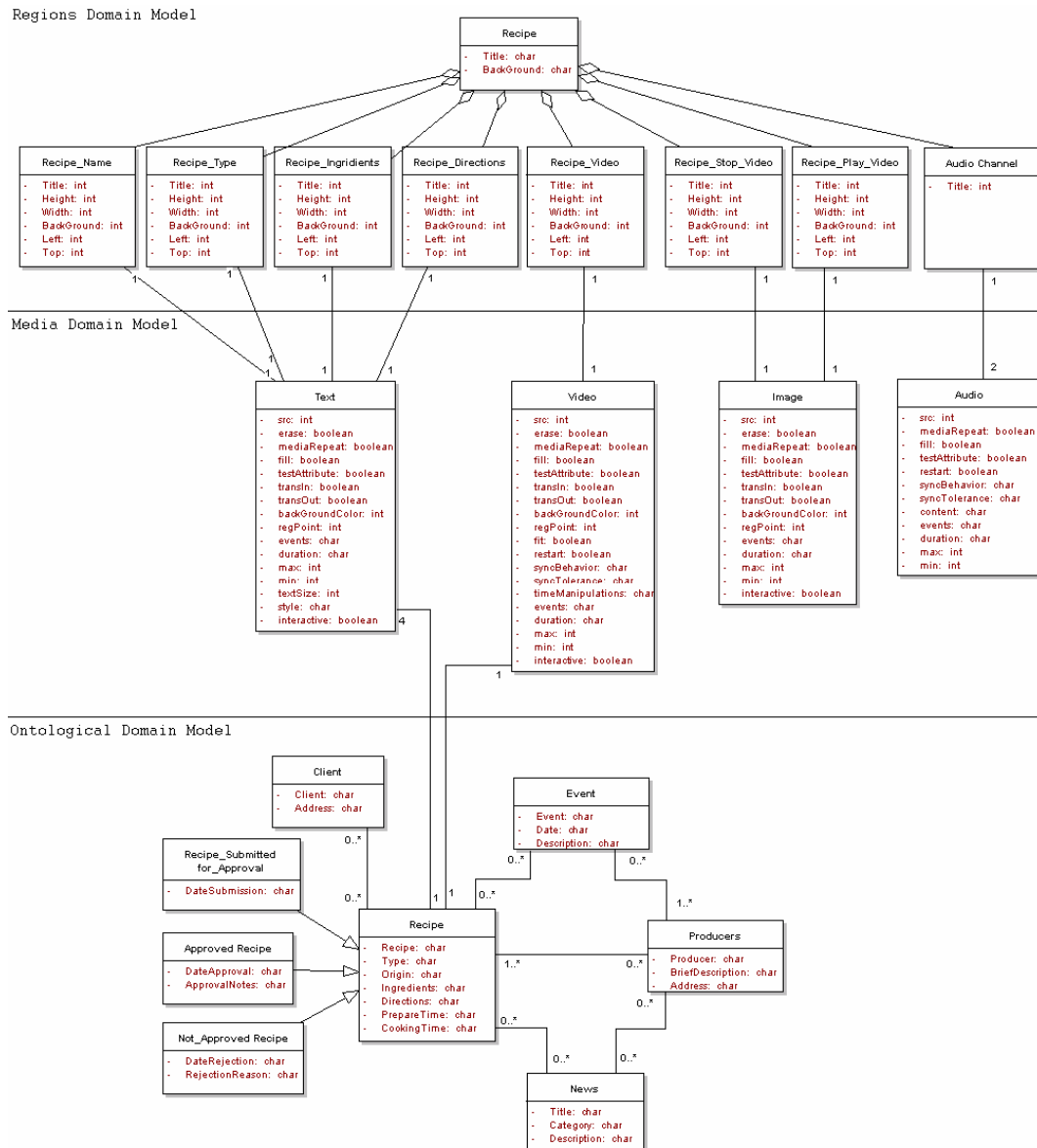


Figure 51: Step 6 - Application Domain Model for the example application.

For instance, consider Figure 51 which illustrates the definition of the class diagram for the description of the *regions domain model*, *media domain model* and *ontological domain model* associated with the previous Application. Notice that in this model the *regions domain model* classes were associated with the *media domain model* classes for the cases when a region will contain one or more media object, for example “Audio Channel” will present two audio media objects and “Recipe Video” will present a video object. A relation was also made between the *ontological domain model* classes and the *media domain model* classes according to the needs of the user interface. Thus, “Recipe” has been associated with the “Video” and “Text” classes once the attributes of “Recipe” will be presented in 4 text media objects and 1 video object.

IV.4.7. Step 7 – Application Object Model (Entities Objects)

The *application object model* is the instantiation of the *application domain model* and it is used to define initial values for the user interface *interaction spaces*, existing user interface *media* and user information. In this last case, it can be used to simulate the *ontological domain model* with possible run-time values.

The *application object model* describes the instantiation of the *application domain model* associated with: *regions domain model*, *media domain model* and *ontological domain model*, resulting respectively in the *regions object model*, *media object model* and *ontological object model* diagrams.

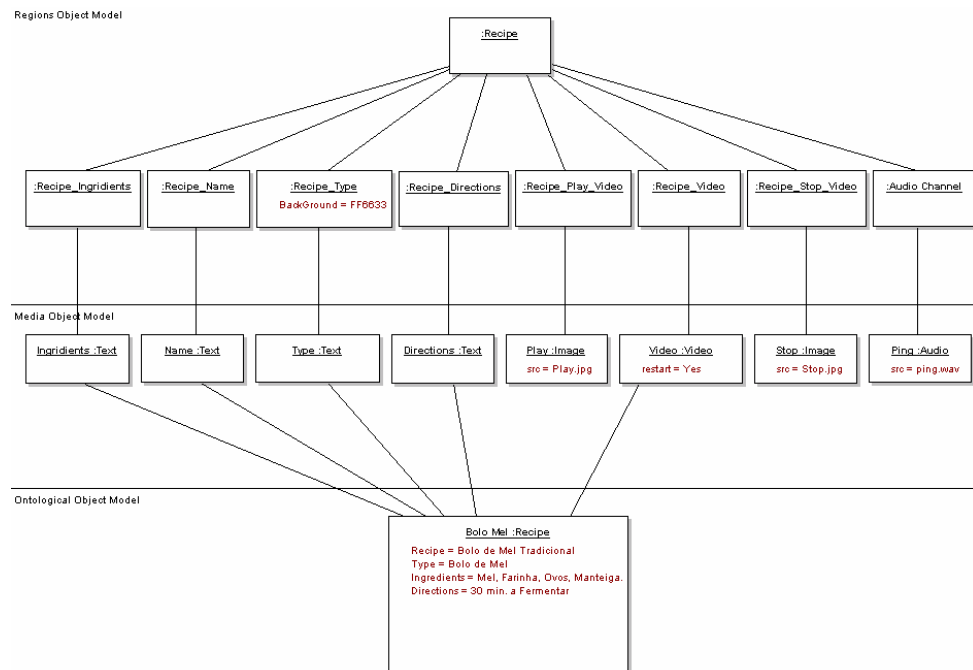


Figure 52: Step 7 - *Object model* for the example application.

All the classes of the *application domain model* can be instantiated in the *application object model* for purposes of value definition and validation of: the existence of the classes; cardinality, name and existence of the relations (among classes); and existence and coherence of the attributes. When classes are instantiated, they assume the run-time values for their attributes allowing for the validation of the class model.

In this model, all classes from the *application domain model* are instantiated if they need value definition or validation. The classes are instantiated, the objects are named and default values are defined for the attributes such as “title” for a region, “src” (source) for a media or any attribute of an object from the *ontological object model*.

In the example presented in Figure 52 some possible run time values were defined according to existing relations from the *application domain model* (Step 6): (i) Source Media for “Recipe_Play_Video”, “Recipe_Stop_Video” and “Audio Channel” regions; (ii) during run-time the existing video (once playing) will restart and repeat itself; (iii) the `backgroundColor` of the “Recipe Type” region will have the value `FF6633` (Orange) and (iv) some example values for “Recipe”.

IV.4.8. Step 8 – Conceptual Model (System Responsibilities + Source)

The *conceptual model* is the diagram that defines both *application business logic* (in the case of IISs) and/or *multimedia behavior* (in the case of IMDs). The same *interaction space* can have more than one associated behavior (a *system responsibility*) that has as a source of information a class or an object (run-time) from the *application domain model* or the *application object model* respectively.

The goal of the *conceptual model* is to associate each *interaction space* or associated user task with a *system responsibility*, and that *system responsibility* with a source of information. Some patterns were tailored for the hybrid functioning of the application in its perspectives of *business logic* and *Multimedia behavior*. As a result, the *conceptual model* aims at associating the products of the methodology so far, i.e., the *presentation model* (visual design of the interface, Step 5) with the *application domain model* (Step 6) and/or *application object model* (Step 7).

Some patterns for the hybrid functioning of the application which combine *business logic* for applications and *Multimedia behavior* are presented in Appendix A.

For the construction of this model, *interaction spaces* and/or *user tasks* of the *presentation model* are related to *system responsibilities*, according to the *interaction model* or existing functional requirements. These *system responsibilities* are then related to a source, i.e., classes or objects depending on what is predicted in the applied pattern.

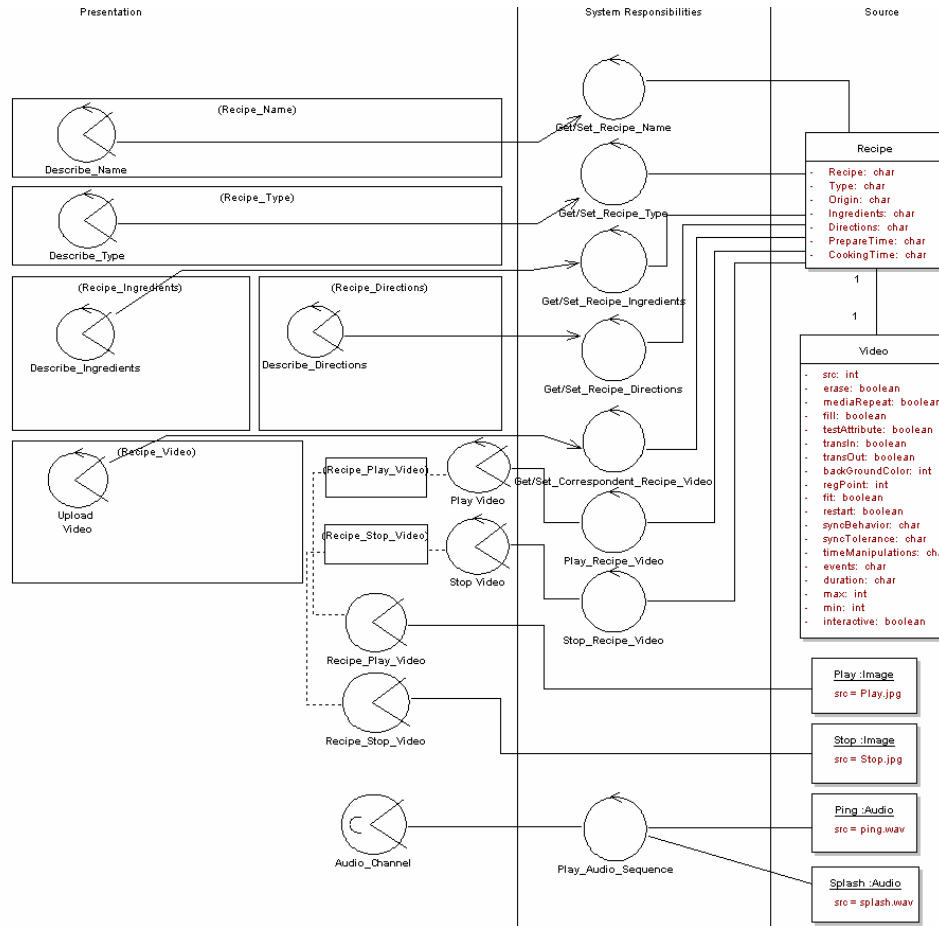


Figure 53: Step 8 - Conceptual model for the example application.

In the example presented in Figure 53 the tasks “Describe Name”, “Describe Type”, “Describe Ingredients”, “Describe Directions” were associated with source by means of the *Get/Set Class Attribute* pattern. The task “Upload Video” is associated with source by means of the *Get/Set Correspondent Media* pattern. The interaction spaces “Recipe Play Video” and “Recipe Stop Video” were associated in a two-tier relation with their source image (*Source Media* pattern). The “Audio Channel” interaction space was associated with their source media “Splash” and “Ping” by means of the *Sequence* pattern.

IV.4.9. Step 9 – System Behavior Model (System Responsibilities)

The *system behavior model* details each *system responsibility* identified until this moment into *sub-system responsibilities* using an *activity diagram*. The *system behavior model* can be especially important to define *Multimedia behavior*, but can also be helpful to detail specific application *business logic* needs, in a sequence of *sub-system responsibilities*.

A *multimedia behavior system responsibility* is defined as a *system responsibility* that has *Present* or *Interrupt* keywords in the begin of the *system responsibility* name. Moreover, the presentation duration of a media object is placed in brackets as follows: [Minimal duration of media, Maximum duration of media]. Notice that the maximum duration of a media can be undetermined which means that the duration is unknown and in this case is represented by $+\infty$.

Activity diagrams follow the classical *activity diagram* as defined in UML [Object Management Group, 2003] except for the exclusive fork which is an extension to the UML's parallel fork. The *multimedia behavior system responsibilities* can be detailed into *sub-system responsibilities* in the following way:

- *Sequence* (one *system responsibility* is executed after the other)

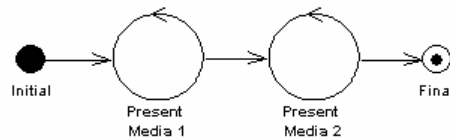


Figure 54: Sequence (of *system responsibilities*).

Figure 54 depicts a sequence, in this kind of structure the *system responsibilities* are executed sequentially, and as a consequence “Media 1” is presented in the first place, and, once its presentation ends “Media 2” is presented.

- *Parallel* (all *system responsibilities* are executed at the same time)

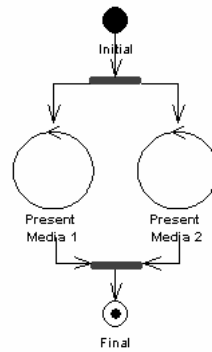


Figure 55: Parallel Fork.

Figure 55 depicts a parallel fork, in this kind of structure, the *system responsibilities* within the fork are executed at the same time and as a consequence “Media 1” and “Media 2” will be presented simultaneously.

- *Exclusive* (all *system responsibilities* are executed but never simultaneously).

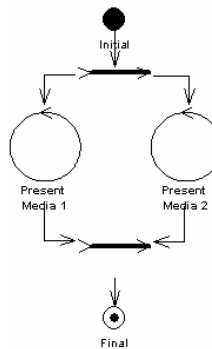


Figure 56: Exclusive Fork.

Figure 56 depicts an exclusive fork, in this kind of structure, the *system responsibilities* within the fork are all executed but not at the same time and as a consequence “Media 1” and “Media 2” will be presented but never simultaneously.

Furthermore, *causal relations* between media objects can also be described. These *relations* assume that a media object presentation can be initiated or interrupted whenever an appropriate event takes place (e.g., start of a media object or a user interaction). Thus, the presentation or interruption of a media presentation can be generated by:

- a user *task* that generates an event over a *multimedia behavior system responsibility*;

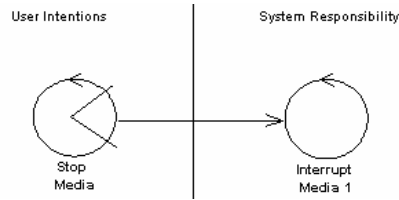


Figure 57: Causal interruption generated from a user interaction.

Figure 57 depicts the representation of a causal relation that was generated from the task “Stop Media” which will have as a consequence the interruption of “Media 1”.

- by the sequence of two *multimedia behavior system responsibilities* in which sequence a *system responsibility* generates an event over another.

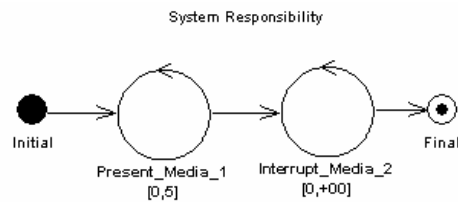


Figure 58: Causal relation generated by the behavior of the system.

Figure 58 depicts the representation of a causal relation generated from the behavior of the system in which the end of the presentation of “Media 1” (that can last between 0 and 5 seconds) will generate the interruption of “Media 2” (which has unknown duration).

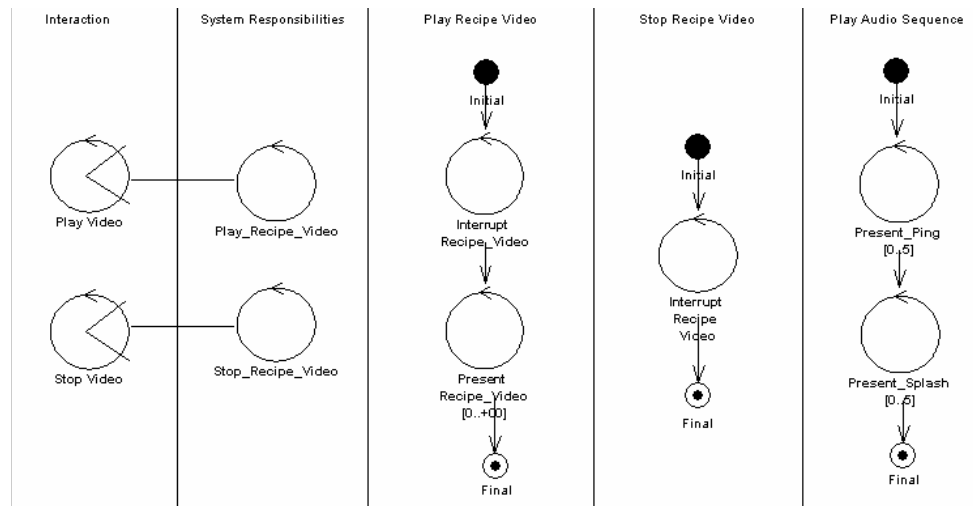


Figure 59: Step 9 - *System behavior model* for the example application.

For instance, consider Figure 59 that illustrates the *system behavior model* for the previous application. The interactions “Play Video” and “Stop Video” generate causal relations over “Play Recipe Video” and “Stop Recipe Video” *system responsibilities* respectively. “Play Recipe Video” is then decomposed into the “Interrupt Video” and “Present Video” (with undetermined duration). The *system responsibility* “Stop Recipe Video” is modeled by only one *system responsibility* producing the interruption of the

video through “Interrupt Recipe Video”. The *system responsibility* “Play Audio Sequence” is implemented by the sequence “Present Ping” and “Present Splash”.

Moreover, notice that using a CASE (Computer Assisted Software Engineering) tool it is possible to represent a composite element which can be further detailed within other (sub-) *activity diagrams*. This is an important potentiality of *MultiGoals*, since it eliminates scalability problems. An alternative view of the (logical/temporal) behavior of the system is the *temporal model* (step 10).

IV.4.10. Step 10 – Temporal Model

The *temporal model* is a time-dependent graphic (timeline) of the *multimedia behavior* of the application. This model also explicits the *causal relations* between the *Multimedia system responsibilities*, whether it was originated by *user tasks* or by *Multimedia logic*. This timeline is described into three different parts:

- *System Responsibilities* (on the upper left side of the graphic depicted in Figure 60) - describes the presentation of all the *multimedia behavior system responsibilities* identified in the *system behavior model* (step 9);
- *Multimedia Logic* (on the upper right side of the graphic) - describes the *Multimedia behavior* of the media objects of the application in time (placed along-side their *multimedia behavior system responsibilities*). For this purpose, this presentation must describe the parallel, sequence and exclusive presentation of all the media objects of the application with their respective presentation duration. Furthermore, the *causal relations* between the media objects can also be described in this part of the graphic by means of a dashed line between the media elements;
- *User Interaction* (on the down right side of the graphic) - describes the possible occurrence of *user interactions* associated with a media object described in the second part of the timeline by means of a dashed line between the *user tasks* and the media.

Moreover, it might also be important to describe on this graphic the *system responsibilities* and their impact on the *presentation* of the *media objects*. *Tasks* and *system responsibilities* that influence the media presentation are placed along the time line from the first moment that they can occur.

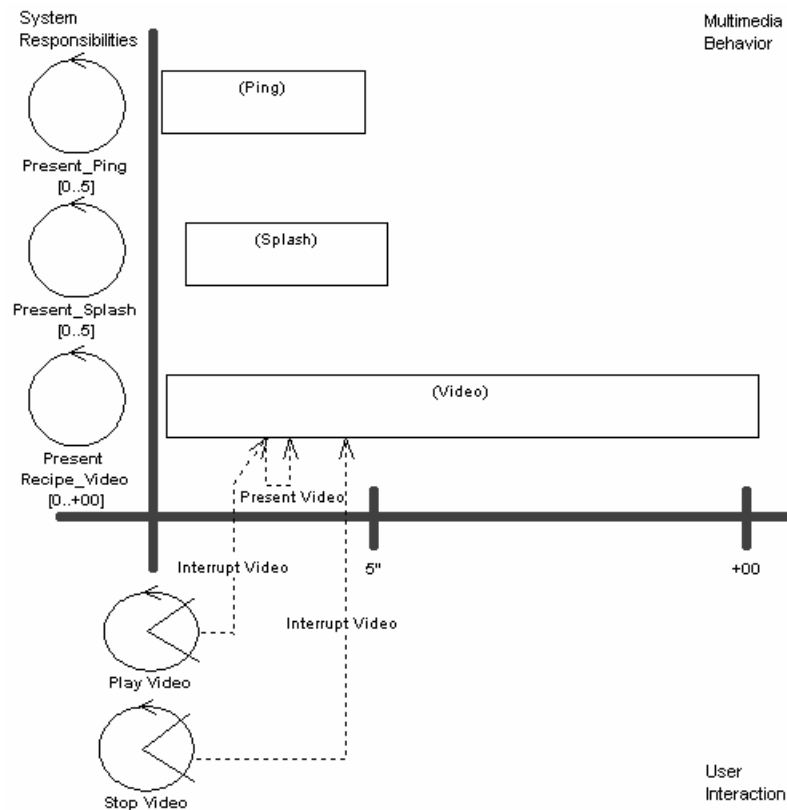


Figure 60: Step 10 - *Temporal model* for the example application.

Consider Figure 60 which depicts the *temporal model*. The *multimedia system responsibilities* responsible for the presentation of media objects are placed in the “System Responsibilities” zone and the corresponding media durations are represented in the “Multimedia Behavior” area where it is possible to identify each *Present* and *Interrupt* functions that influence the presentation of elements. The user tasks “Play Video” and “Stop Video” placed in the “User Interaction” area generate causal relations that affect the video presentation.

IV.4.11. Step 11 – Application Architecture

The *application architecture* is the representation of all the relevant *components* of the *system* and the *relations* among them. The structure of these *components* (which are relevant for the implementation) is organized from left to right (see Figure 61). These components are: *interaction spaces*, *tasks*, *system responsibilities* and *source*.

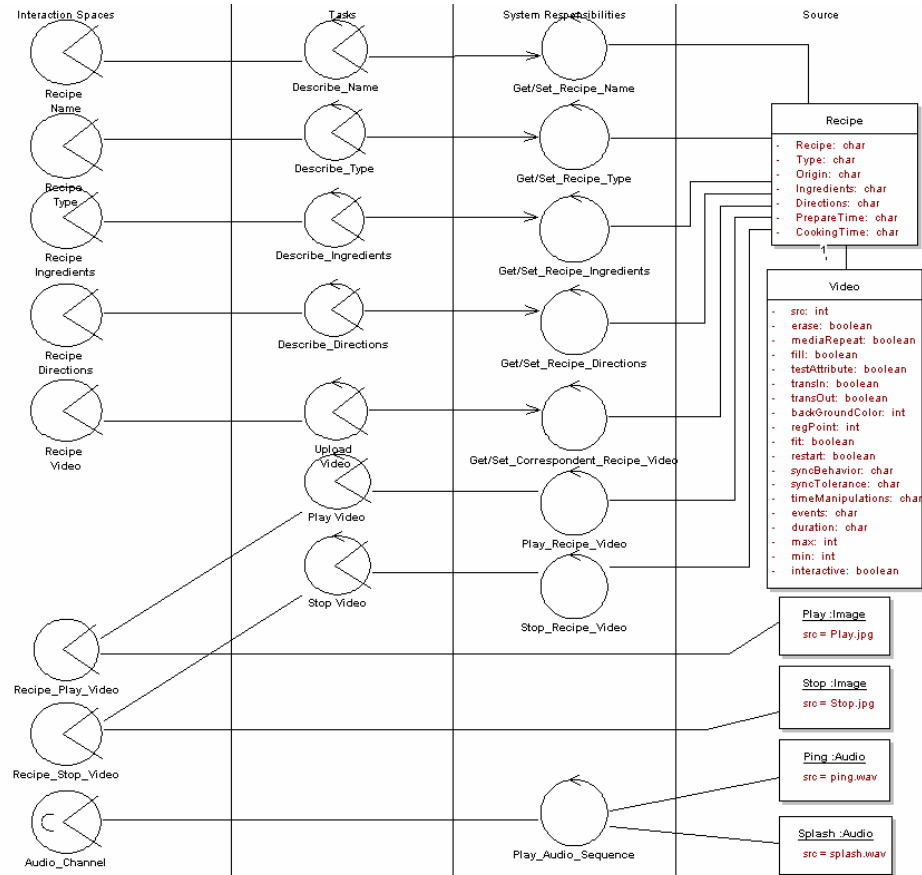


Figure 61: Step 11 - Application architecture for the example application.

An *architecture* of the system is essential to evaluate the system size (and complexity) and to control the system implementation, since it is possible to identify the precedence of implementation between components, e.g. in order to be able to implement the controls “Get/Set Recipe Name” and “Get/Set Recipe Ingredients” the source “Recipe” must already be available.

Furthermore, the architecture is an overall documentation of the system that encourages dialogue between system stakeholders (e.g. client and developing team) in order to reach negotiation over, for instance, the implementation project or system maintenance for the introduction of new requirements.

IV.5. CONCLUSIONS

This work presents *MultiGoals*, a methodology that aims to bridge Software Engineering and Multimedia authoring. *MultiGoals* is a lightweight, *use case*-driven, architectural centric methodology that guides the software conceptualization by means of the simplification of the system design concerning usability and maintainability.

The use of *MultiGoals* induces the author of an application into a straight lined (few iterations are suggested) and fast software definition process towards implementation. The models of the methodology are intended to be simple, intuitive and scalable. A simple example is used to illustrate how the methodology can be applied, however, *MultiGoals* can be used to develop complex applications.

Despite the proposal of 11 steps for the design of Interactive Information Systems, these steps support completely the design of complex applications. However, as a matter of simplicity, many of these steps can be omitted in order to achieve faster products of the methodology.

V. CASE STUDY (MULTIGOALS)

The previous chapters presented the *Process Use Cases* and *MultiGoals* methodologies. However, the examples provided to support the presentation were applied by the author of the methodologies and as a result have a considerable academic weight. For this reason was important to test the methodologies in a real environment so that the training and the modeling could be evaluated.

Once *Process Use Cases* had already been applied several times (it has been used for around two years in the University of Madeira), it was found that the focus should be on *MultiGoals*. Due to human resources knowledge and time restrictions it was only possible to apply the *MultiGoals* simplified version which is presented in the sequel.

V.1. INTRODUCTION

This chapter presents the application of the simplified version of *MultiGoals*. To present this case study we apply the Gastronomy Project that was previously used for the presentation of the *Process Use Cases* and *MultiGoals* methodologies in chapters III (which presents the project comprehensively) and IV, respectively.

As presented in the previous chapter, the *MultiGoals* simplified version should be applied when the nature of the problem to be solved is relatively simple. The Gastronomy Project presented in this thesis fits this approach since no complex system behavior is predicted and the Multimedia requirements are kept to a minimum. Another advantage of the simplified version is that it can be used by persons with no training on Software Engineering methods like usually happens with Multimedia designers.

The simplified version of *MultiGoals* methodology was applied by two Multimedia designers. These designers - Filipe Freitas [Filipe Freitas, 2007] and Paulo Vieira [Paulo Vieira, 2007] - work daily on digital Multimedia producing logos, pamphlets, animations and web sites using tools like ®Adobe Photoshop [Adobe, 2007b] and ®Adobe Flash [Adobe, 2007a]. The designers where hired to participate in the Gastronomy Project to partially model the application and to develop the designed system interface.

In order to apply the methodology, a 2 hour training session took place, in which the Gastronomy Project was presented using the material illustrated along the Chapter III - Requirements. After the project was introduced, the training focused on the simplified version of *MultiGoals* (models: 1 - *Use cases model*; 2 - *Activity diagrams*; 3 - *Interaction model*; and 5 - *Presentation model*) using the material illustrated along the Chapter IV - Analysis and Design.

After the training session (October 16th 2007), the two Multimedia designers were able to model by themselves (October 30th 2007) the diagrams that are presented in this chapter. A final revision of the models was made (November 14th 2007) where a final iteration of the process was carried out. Finally, the produced user interfaces were redrawn with digital support (November 25th 2007) and presented to the client for

approval (November 27th 2007). Each produced model is commented and for each one the relevant conclusions for the evolution of the methodology are presented.

This chapter is organized as follows: Section V.1 presents the training session of the *MultiGoals* simplified version, Section V.2 presents the diagrams that were produced for the Gastronomy Project, and Section V.3 presents some conclusions about the results of the application of the methodology.

V.2. TRAINING

The *MultiGoals* training had the purpose of providing the two Multimedia designers with the sufficient knowledge to make (without help) the analysis and design of the Gastronomy Project based on the *use cases* previously identified in Chapter III, by the application of the *Process Use Cases* methodology: “Catalog Recipe” (to be remodeled during the training session); “Advertise” and “Obtain Gastronomic Information”. Due to time constraints, the training session had to be scheduled (Table 5) for only two hours.

Table 5: *MultiGoals* training session schedule.

Topic	Duration (m)	Begin (m)	End (m)	Content
Objectives	5	0	5	Learn to model applications using <i>MultiGoals</i> .
Context	5	5	10	Software Engineering Basics: Requirements, Analysis and Design.
Project	20	10	30	Project Interiorization using Process Use Cases models (Chapter III).
Use Cases	30	30	60	What is a use case? (Model 1 – Use Cases) Activity Diagrams (Model 2 – Activity Diagrams)
Interaction Model	30	60	90	Task decomposition using CTT and association to System Responsibilities (Model 3 – Interaction Model)
Presentation	30	90	120	Presentation Modeling and task association (Model 5 – Presentation Model).

The session, as depicted in the previous table, included the following topics:

- Objectives – it was explained the objective of the training session as to learn how to model Interactive Information Systems using the *MultiGoals* simplified version;
- Context – it was explained a Software Engineering process having several phases and, that the designers would work on the phases of analysis and design following the already completed phase of requirements;
- Project – the Gastronomy Project was presented using the models introduced in Chapter III focusing on the *high-level concept*, the identification of business processes and their design (*process use cases model*);
- Use Cases – it was explained what an *essential use case* is, how it can be decomposed into user intentions and *system responsibilities* using an *activity diagram*, and how an *interaction space* could be associated with each *task*;

- Interaction Model – it was explained how a pair *task-system responsibility* can be decomposed into sub-*tasks* (using CTTs) and sub-*system responsibilities*;
- Presentation – it was presented how the user interface could be designed and how each *interaction space* should be associated with each user *task*.

Following the training session, an agreement was made with the two Multimedia designers that they would complete the modeling of the system within 15 days (until the end of October).

The next section presents the diagrams that were produced in the different sessions of modeling until the final client approval.

V.3. PRODUCED DIAGRAMS

In this section we present the diagrams produced for the design of the system for the Gastronomy Project. For each diagram (*use cases model*, *activity diagrams*, *interaction model*, and *presentation model*), we present the title, the author(s), the date, a description of the diagram, and brief conclusions aiming at the evolution of the *MultiGoals* methodology.

The diagrams are presented in its original, digitalized paper version in order to preserve all the information produced e.g. handwritten notes. To ease the understanding of the diagrams presented, we provide pointers (in orange) to the diagrams classes so that text and figures can be easily related.

V.3.1. Step 1 - Use Cases

The first diagram was produced just after the training session and indicated the *use cases* that needed to be analyzed.

Use Cases diagram

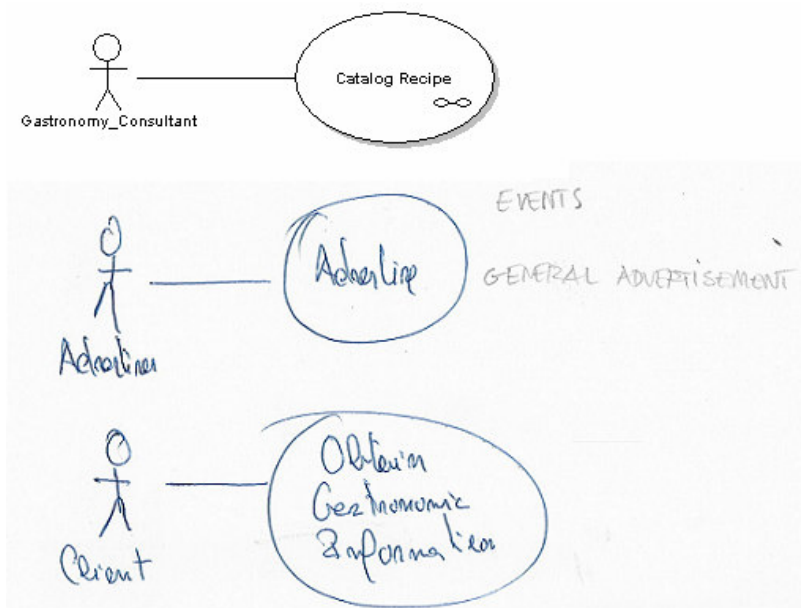


Figure 62: Step 1 - *Use cases* of the Gastronomy Project.

Author(s): Pedro Valente

Date: October 16th 2007

Description: The diagram presents the *use cases* of the Gastronomy Project (Figure 62). The “Advertise” and “Obtain Gastronomic Information” *use cases* were designed by hand and the “Catalog Recipe” was copy-pasted in order to complete the *use cases* diagram. Some informal annotations were made in the diagram by Filipe Freitas and Paulo Vieira (“Events” and “General Advertisement”)

Following the presentation of the *use cases model*, we now present the *activity diagrams* related to each identified *use case*.

V.3.2. Step 2 - Activity Diagram

Following the definition of the *use cases* we now present the *activity diagrams* for the “Catalog Recipe”, “Advertise” and “Obtain Gastronomic Information” *use cases*.

“Catalog Recipe” Activity Diagram

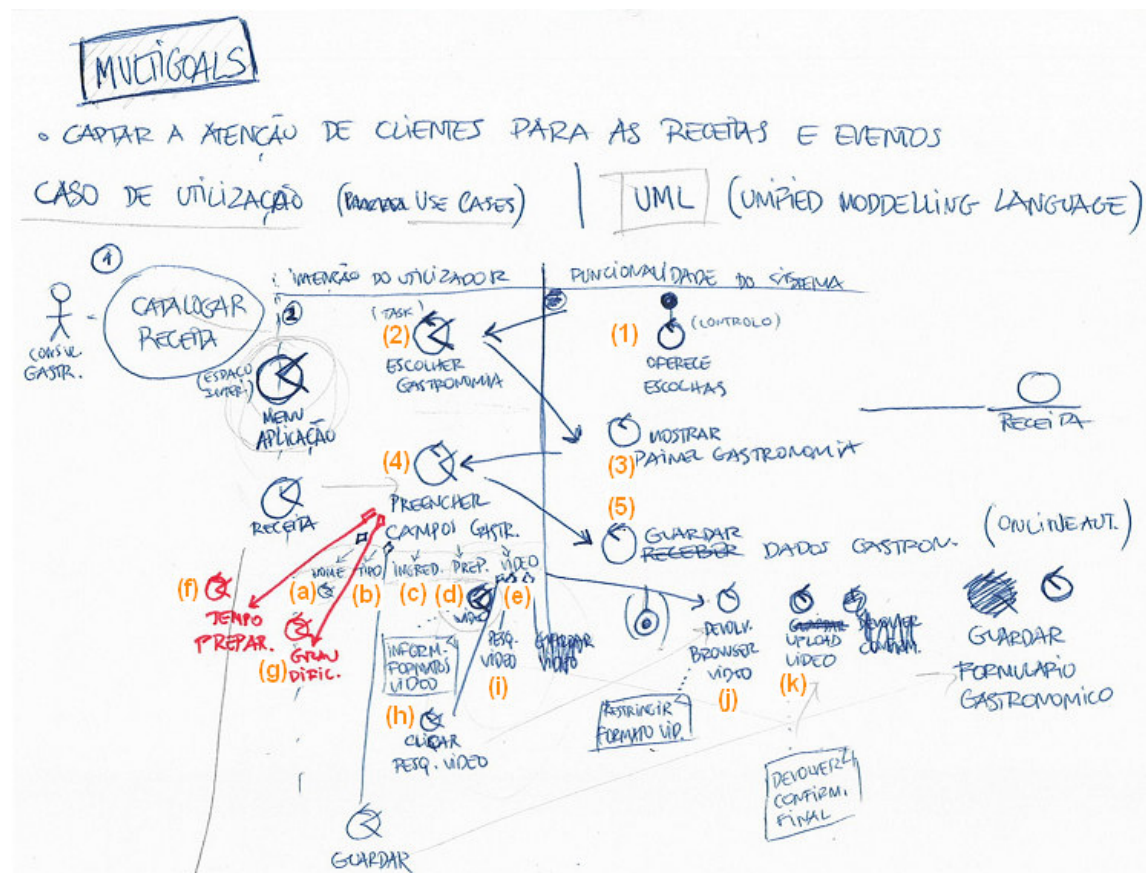


Figure 63: Step 2 - Activity diagram for the "Catalog Recipe" use case.

Author(s): Filipe Freitas and Paulo Vieira (Version 1); Filipe Freitas, Paulo Vieira and Pedro Valente (Version 2).

Date: October 16th 2007 (Version 1, during the training session), November 14th 2007 (Version 2).

Description: The first version of this diagram (Figure 63) starts with the system “Offering Choices” (“Oferece Escolhas”)(1), then the user “Choose(s) Gastronomy” (“Escolher Gastronomia”)(2) then the system responds “Show(ing) Gastronomy Panel” (“Mostrar Painel Gastronomia”)(3), after that the user “Fill(s) the Gastronomy Fields” (“Preencher Campos Gastr.”)(4), and finally the system “Keeps the Gastronomy Data” (“Guardar Dados Gastron.”)(5).

By analyzing the diagram, we notice that the “Fill the Gastronomy Fields” (4) *task* was decomposed into the *tasks*: “Name” (“Nome”)(a); “Type” (“Tipo”)(b); “Ingredients” (“Ingred.”)(c); “Directions” (“Prep.”)(d), “Video” (e), “Preparation Time” (“Tempo Prepar.”)(f), and “Difficulty Level” (“Grau Dific.”)(g).

The *task* “Video” was still decomposed into the *tasks*: “Click Video Search” (“Clicar Pesq. Video”)(h); and “Search Video” (“Pesq. Video”)(i); and it was also associated with the following *system responsibilities*: “Return Video Browser” (“Devolv. Browser Video”)(j); and “Upload Video” (k). This premature decomposition of the “Fill the Gastronomy Fields” *task* was a training error that was not detected during the session that would influence the modeling of the *activity diagrams* for the remaining *use cases*.

The following *interaction spaces* were identified: “Application Menu” (“Menu Aplicação”) and “Recipe” (“Receita”).

Conclusions: The premature decomposition of the “Fill the Gastronomy Fields” *task* was made in the *activity diagram* by mistake. This *task* should only be decomposed during the Step 3 - *Interaction Model*, however, during the training session the two Multimedia designers drawn this decomposition on the same diagram. When questioned about this mistake, they answered that they thought that this decomposition should be made in a different diagram due to whiteboard space restrictions. However, even resulting from a mistake this approach should be taken into account for being a “two (diagrams) in one”, and since the same cognitive steps are carried out, the result of that interpretation of the system structuring is being accomplished. However, this mistake led to a much more complex diagram.

“Advertise” Activity Diagram

Author(s): Filipe Freitas and Paulo Vieira.

Date: October 30th 2007.

Description: The diagram depicted in Figure 64 starts with the system “Show(ing) News/Event/Publicity Panel” (“Mostrar Painel Noticia/Evento/Publicidade”)(1), then the user “Fill(s) the Fields” (“Preencher Campos”)(2) which was decomposed

into the tasks "Title"("Título"), "News"("Notícia"), "Media", "Link" and "Save"("Guardar"). The task "Media"(3) is further decomposed into "Click Search" ("Clicar Pesquisar") and "Search Media" ("Pesquisar Media"). The "Media" task is associated with the "Return Image Browser" ("Devolver Browser Imag.") and "Upload Image" ("Upload Imagem"). Following the task "Save" ("Guardar")(4), the system responds "Saving the Data"("Guardar Dados")(5).

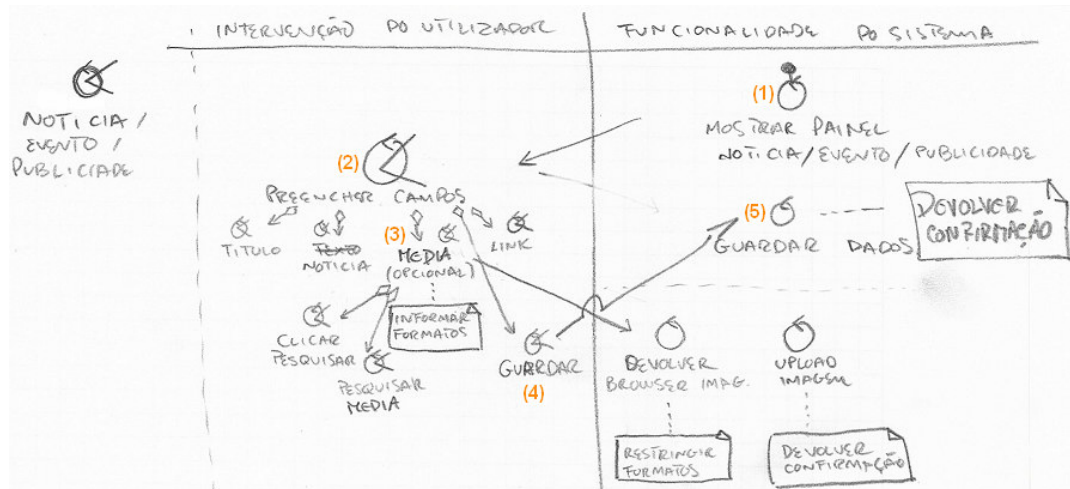


Figure 64: Step 2 - Activity diagram for the "Advertise" use case.

The following interaction space was identified: "News/Event/Publicity" ("Notícias/Evento/Publicidade").

Conclusions: Following the previous diagram, the same task decomposition was made in the activity diagram and it has resulted in a more confusing and difficult to understand diagram. However, the objective of the Step 3 - Interaction Model was accomplished since two extra system responsibilities were identified: "Return Image Browser" ("Devolver Browser Imag.") and "Upload Image" ("Upload Imagem").

"Obtain Gastronomic Information" Activity Diagram

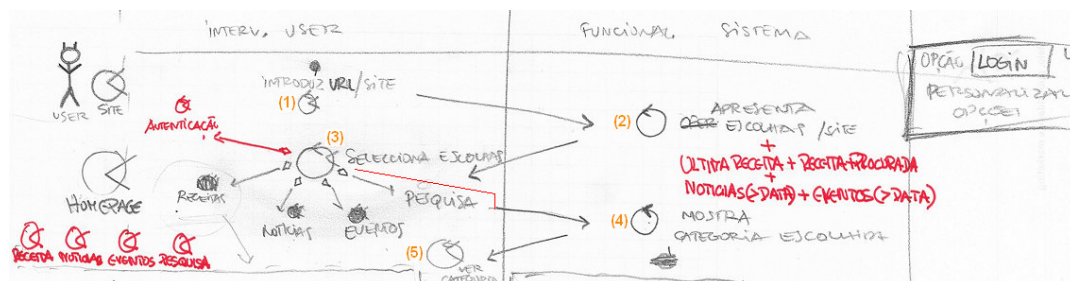


Figure 65: Step 2 - Activity diagram for the "Obtain Gastronomic Information" use case.

Author(s): Filipe Freitas and Paulo Vieira (Version 1), Filipe Freitas, Paulo Vieira and Pedro Valente (Version 2).

Date: October 30th 2007 (Version 1, during the training session), November 14th 2007 (Version 2).

Description: The diagram (Figure 65) starts with the user “Introduc(ing) the URL/Site” (“Introduz URL/Site”)(1), the system responds “Presenting the site choices+(plus) Last Recipe+More Searched Recipe+News (ordered descending by date) + Events (ordered descending by date)” (“Apresenta Escolhas Site+Ultima Receita + Receita+Procurada + Notícias (>Data)+Eventos (>Data)”)(2), then the user “Select(s) one of the Choices” (“Selecciona Escolhas”)(3), and the system responds “Return(ing) the Chosen Category” (“Mostra Categoria Escolhida”)(4). Finally, the user “Watches the Category” (“Ver Categoria”)(5). The “Select(s) one of the Choices” task is further decomposed into “Recipe” (“Receitas”), “News” (“Notícias”), “Events” (“Eventos”), “Search” (“Pesquisa”) and “Authentication” (“Autenticação”). At the top right of the diagram there is a draft for the *interaction space* of the “Authentication” task.

The following *interaction spaces* were identified: “Site” and “HomePage”.

Conclusion: This diagram follows the method already applied in the previous diagrams.

V.3.3. Step 3 – Interaction Model

As it was possible to conclude in the previous section, there was a mistake and the steps 2 and 3 of the methodology were drawn in a single diagram. However, during the modeling of the diagrams, the two Multimedia designers needed to further decompose existing *tasks* and, for that purpose, used the same principle and decomposed their *tasks* always associating them with *system responsibilities* which is the principle applied in the Step 3 – Interaction Model.

“Select one of the Choices”- “Return the Chosen Category” (Recipe Situation)

This *interaction model* is related to the pair *task-system responsibility* “Select one of the Choices” (“Selecciona Escolhas”)-“Return the Chosen Category” (“Mostra Categoria Escolhida”) of the “Obtain Gastronomic Information” *activity diagram*. Since the task “Select one of the Choices” (“Selecciona Escolhas”) – Step 3 of the diagram depicted in Figure 65 - can be decomposed into several sub-*tasks*, the two Multimedia designers have chosen to draw the diagram for the “Recipe” situation.

Author(s): Filipe Freitas and Paulo Vieira (Version 1).

Date: October 30th 2007.

Description: The diagram depicted in Figure 66 starts with the user *task* “Select Recipe Menu” (“Escolher Menu Receitas”)(1) and the system responds with the “Show(ing) the Recipes” (“Mostra Receitas”)(2) *system responsibility* which is further decomposed into “More Recent Recipes” (“Receitas + Recentes”) and “Recipes by Type” (“Receitas por Tipo”). Then, the user “Select(s) Recipe from the List” (“Selecciona Receita da Lista”)(3) and the system responds “Show(ing) the Recipe” (“Mostra a Receita”)(4), finally the user “Visualize(s) the Recipe” (“Visualiza Receita”)(5). This last *task* “Visualize Recipe” is further decomposed into “Download”, “Print” (“Imprimir”), “Send Mail” (“Enviar Mail”), “Save in Site” (“Guardar no Site”) and “Watch Media” (“Ver Media”) which is further decomposed into “Save Video to Disk” (“Guardar Video no Disco”).

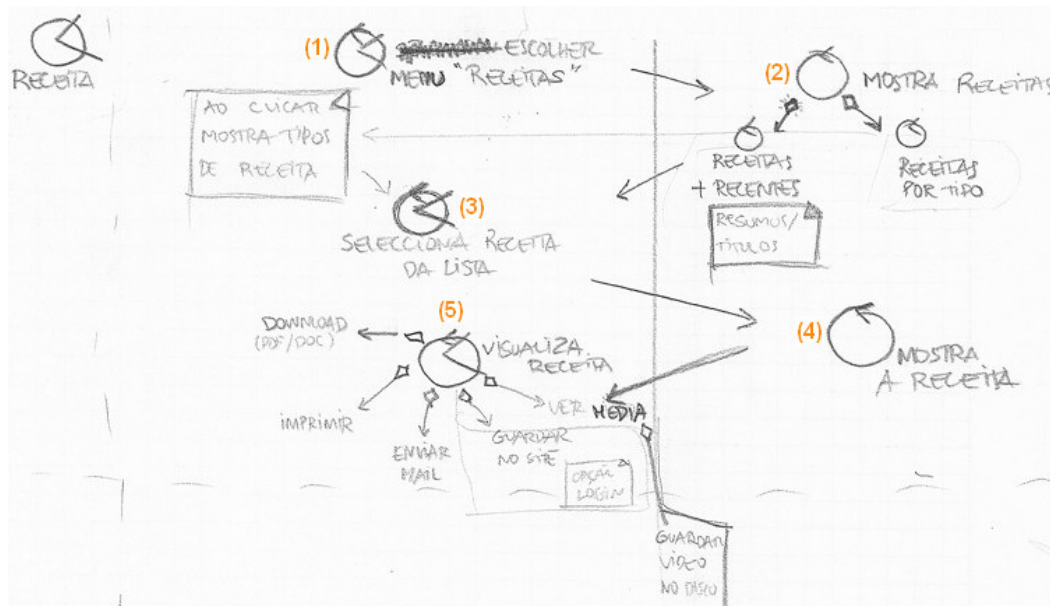


Figure 66: Step 3 - Interaction model for the pair task-system responsibility “Select one of the Choices” - “Return the Chosen Category” (Recipe Situation).

The following *interaction space* was identified: “Recipe” (“Receita”), which was already previously identified.

Conclusion: The *interaction model* was incorrectly drawn once it was designed just like the previous *activity diagrams*. However, the objectives of the diagram were accomplished since a further iteration was made on the detail of user intentions and *system responsibilities*. There is the need to study a solution to aggregate all the information resulting from this diagram into the *interaction model*. The choice to model the “Recipe” situation was a correct one, once the Recipes are the more important issue of the application. By analyzing the diagram, we can note that one *interaction space* is missing, the “Recipe List”, in which the *task* “Select Recipe from the List” (“Selecciona Receita da Lista”) would take place. As a result, this *interaction space* was not designed.

“Select one of the Choices”- “Return the Chosen Category” (Search Situation)

Following the previous decomposition, the need to decompose the “Search” situation was noticed. This diagram was made with the monitoring of the author of this thesis and was an attempt to give a different format to the model, making the decomposition of the *tasks* from left-to-right instead of top-to-bottom.

Author(s): Filipe Freitas, Paulo Vieira and Pedro Valente.

Date: November 14th 2007.

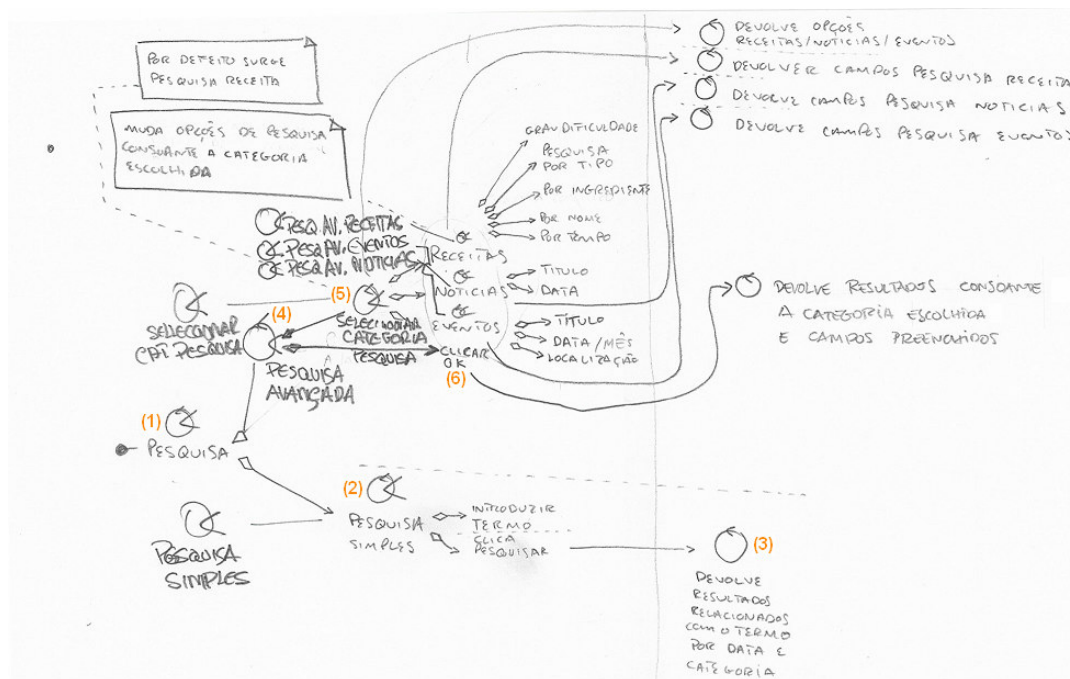


Figure 67: Step 3 - Interaction model for the pair task-system responsibility “Select one of the Choices”- “Return the Chosen Category” (Search Situation)

Description: The diagram depicted in Figure 67 “Select one of the Choices”- “Return the Chosen Category” starts with the *task* “Search”(“Pesquisa”)(1) which is decomposed into:

- “Simple Search”(“Pesquisa Simples”)(2) that is further decomposed into “Introduce Term” (“Introduzir Termo”) and “Click Search” (“Clica Pesquisar”) and the system responds “Returning results related to the term ordered by date and category” (“Devolve Resultados Relacionados com o Termo por Data e Categoria”)(3);
- “Advanced Search”(“Pesquisa Avançada”)(4) which is further decomposed into “Select Category of Search”(“Seleciona Categoria Pesquisa”)(5) and “Click OK”(“Clicar OK”)(6). “Select Category of Search” is further detailed into “Recipes”(“Receitas”), “News”(“Notícias”), “Events”(“Eventos”). The categories of search were further detailed into:

- “Recipes” - “Difficulty Level” (“Grau Dificuldade”); “Type” (“Pesquisa por Tipo”); “Ingredient” (“Por Ingrediente”); “Name” (“Por Nome”); and “Time” (“Por Tempo”);
- “News” - “Title” (“Título”) and “Date” (“Data”);
- “Events” - “Title” (“Título”) and “Date/Month” (“Data/Mês”) and “Location” (“Localização”).

The following *tasks* were associated with the following *system responsibilities*:

- “Select Category of Search” was associated with “Return Recipe/News/Events Options” (“Devolve Opções Receitas/Notícias/Eventos”);
- “Recipes” was associated with “Return Fields of Recipe Search” (“Devolver Campos Pesquisa Receita”);
- “News” was associated with “Return Fields of News Search” (“Devolver Campos Pesquisa Notícias”);
- “Events” was associated with “Return Fields of Events Search” (“Devolver Campos Pesquisa Eventos”);
- “Click OK” was associated with “Return Results following the selected category and filled fields” (“Devolve Resultados Consoante a Categoria Escolhida e Campos Preenchidos”).

The following *interaction spaces* were identified: “Simple Search” (“Pesquisa Simples”); “Select Search Category” (“Seleccionar Cat. Pesquisa”); “Recipe Advanced Search” (“Pesq. Av. Receitas”); “Events Advanced Search” (“Pesq. Av. Eventos”); and “News Advanced Search” (“Pesq. Av. Notícias”).

Conclusions: This diagram was drawn according to the Step 3 – Interaction Model. However, by the analysis of the diagram, it can be concluded that the resulting diagram is complex when there are many decompositions of the identified *tasks*. Although, it is natural that a diagram turns out to be complex when there is a complex problem to be solved, it should be identified a solution to generate less confusing diagrams.

V.3.4. Step 5 – Presentation Model

As a result of the modeling of the previous diagrams, the following *interaction spaces* (ISs) were identified: “Recipe”; “News/Events/Publicity”; “Site”; “HomePage”; “Application Menu”; “Simple Search”; “Select Search Category”; “Recipe Advanced Search”; “Events Advanced Search”, and; “News Advanced Search”. The *presentation*

model defines the spatial relations of the user interface objects. These ISs are respectively presented along this section.

“Recipe”

Author(s): Filipe Freitas and Paulo Vieira (version 1, paper support blue ink); Filipe Freitas, Paulo Vieira and Pedro Valente (version 2, paper support red ink); Pedro Valente (version 2.1, digital support); João Dionísio (the client) and Pedro Valente (Version 3, digital support with manuscript red ink).

Date: October 16th 2007 (version 1), November 14th 2007 (version 2) November 25th 2007 (version 2.1); November 27th 2007 (version 3).

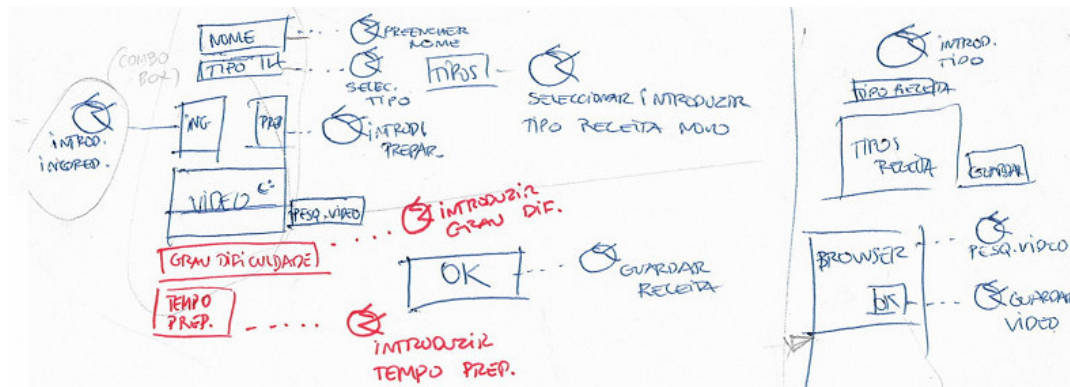


Figure 68: Step 5 - "Recipe" IS versions 1 (blue ink) and 2 (red ink).

Description: The first version of the “Recipe” IS (Figure 68) is composed of the following objects: “Name” (“Nome”); “Type” (“Tipo”); “Ingredients” (“Ing.”); “Directions” (“Prep.”); “Video”; “Video Search” (“Pesq. Video”) and “OK” which are described as buttons. The second version of the IS added the objects: “Difficulty Degree” (“Grau Dificuldade”) and “Preparation Time” (“Tempo Prep.”). Each object of the IS is associated with the *task* which is performed on it. On the right side of the diagram, separated by a blue vertical line is the design of the ISs whose main purpose is: (upper right side) the introduction of the recipe types, and; (lower right side) browse of the media for the recipe.

The version 2.1 of the IS (Figure 69) is a copy of the version 2 except that the “Play” and “Stop” buttons were added. This version was presented to the client for approval, from that, a third version was generated in which the “Source” and “Recipe History” fields were added to complete the final and approved version of the IS.

Recipe Name	Recipe Type
Ingredients	Directions
Media	Difficulty Level
	Preparation Time (Minutes)
	Source: Cook traditions in Winton
	Recipe History
Play	Stop
(Search Media)	Save

Figure 69: Step 5 - "Recipe" IS versions 2.1 and 3 (red ink, approved by the Client).

Conclusions: The association of the user *tasks* with the objects of the IS (version 1) might have contributed for the identification of the two (extra) ISs for the introduction of the recipe types and media browser. This is assumed once the modeler is guided to extensively think about the *tasks* that the user must accomplish in the IS that is being designed.

“News / Event / Advertisement”

Author(s): Filipe Freitas and Paulo Vieira (version 1, paper support pencil); Filipe Freitas, Paulo Vieira and Pedro Valente (version 2, paper support red ink); Pedro Valente (version 2.1, digital support).

Date: October 30th 2007 (version 1), November 14th 2007 (version 2) November 25th 2007 (version 2.1).

Description: The first version of the “News/Event/Publicity” IS (Figure 70) is composed of the following objects: “Title”(“Título”); “Text”(“Texto”); “Search Media”(“Pesq. Media”) which is a button; “Link”, and; “OK” which is also a button. The second version of the IS introduced the “Location” (“Localização”), “Date” (“Data”) and “Keywords” (“Pal. Chave”) fields. Each object of the IS is associated with the *task* which is performed by it. On the right side of the diagram is the design of the ISs to browse the media for the advertisement.

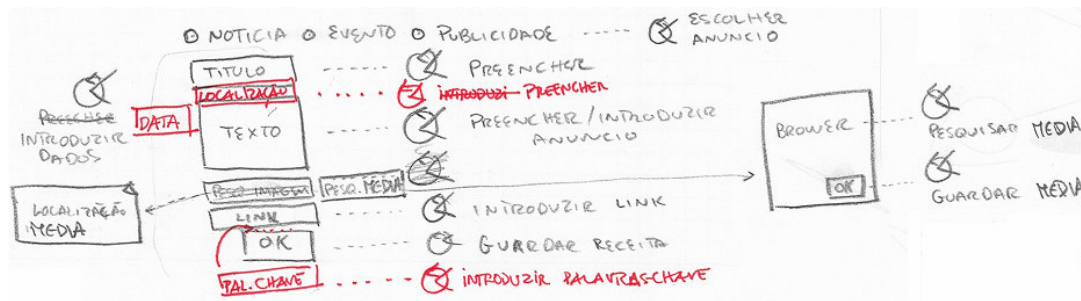


Figure 70: Step 5 - "Advertise" IS versions 1 (pencil) and 2 (red ink).

Advertisement Title	News / Event / Publicity
Advertisement Description	Date
	Location
	Link (URL)
	KeyWords
Media	
Play	Stop
Search Media	Save

Figure 71: Step 5 - "News/Event/Publicity" IS version 2.1 (Approved by the Client).

The version 2.1 of the IS (Figure 71) is a copy of the version 2 except that the "Play" and "Stop" buttons were added. This version was presented to the client and approved without changes.

"Site", "HomePage", "Application Menu", "Simple Search"

Author(s): Filipe Freitas and Paulo Vieira (version 1, paper support pencil); Pedro Valente (version 1.1, digital support); João Dionísio (the client) and Pedro Valente (Version 2, digital support with manuscript red ink).

Date: October 30th 2007 (version 1), November 25th 2007 (version 1.1), November 27th 2007 (version 2).

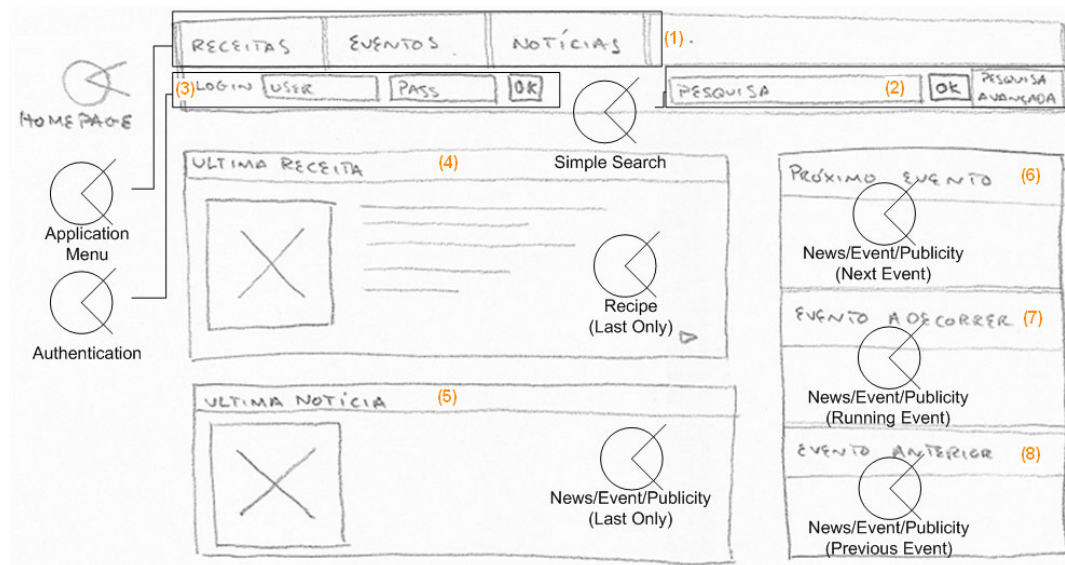


Figure 72 : Step 5 - "HomePage" IS version 1.

Description: During the modeling of the “Obtain Gastronomic Information” *activity diagram* an IS named “Site” was identified. However, when questioned about the meaning of this IS, the two Multimedia professionals answered that it was related to the web browser (where the URL was typed). For this reason, the IS does not need representation. This “HomePage” IS is closely related to the “Obtain Gastronomic Information” *activity diagram* in which the user *tasks* “Recipes”, “News”, “Events”, “Search” and “Authentication” were identified. Based of the information acquired in this diagram, the IS is composed of the following sub-ISs:

- “Application Menu” IS (1), which is composed of the buttons “Recipes” (“Receitas”), “Events” (“Eventos”) and “News” (“Notícias”) to support the *tasks* “Recipes”, “Events” and “News”;
- “Simple Search” IS (not formally identified previously)(2) which is the button that leads the user to the “Advanced Search” that supports the *task* “Search”;
- “Authentication” IS (not formally identified previously)(3) of the user which is in the fields “User” and “Pass(word)” and supports the *task* “Authentication”;
- “Last Recipe” (“Última Receita”) IS (not formally identified previously)(4) that is the “Recipe” IS in a read-only mode filtered by the last recipe introduced in the system;
- “Last News” (“Última Notícia”) IS (not formally identified previously)(5) which is the “News/Event/Publicity” IS in a read-only mode filtered by the last news introduced in the system;
- “Next Event” (“Próximo Evento”) IS (not formally identified previously)(6) which is the “News/Event/Publicity” IS in a read-only mode filtered by the next event;

- “On-Going Event” (“Evento a Decorrer”) IS (not formally identified previously)(7) which is the “News/Event/Publicity” IS in a read-only mode filtered by the event that is running, and;
- “Last Event” (“Evento Anterior”) IS (not formally identified previously)(8) which is the “News/Event/Publicity” IS in a read-only mode filtered by the last event.

Figure 73: Step 5 - "HomePage" IS versions 1.1 and 2 (red ink, approved by the Client).

The version 1.1 of the IS (Figure 73) is a copy of the version 1. This version was presented to the client and a second version of the IS was generated, since the Client only wanted to show the fields “Recipe Name”, “Recipe Type” and “Recipe History” of the “Recipe” read-only IS.

Conclusions: Although the majority of the ISs that compose the “HomePage” IS were not formally identified in the previous diagrams, the detailed analysis of the user *tasks* led/enabled the identification of 6 ISs that can help the user in existing user *tasks*.

“Select Search Category”, “Recipe Advanced Search”, “Events Advanced Search”, “News Advanced Search”

Author(s): Filipe Freitas, Paulo Vieira and Pedro Valente (version 1, paper support pencil); Pedro Valente (version 1.1, digital support).

Date: November 14th 2007 (version 1), November 25th 2007 (version 1.1).

The image shows three hand-drawn wireframes for an 'Advanced Search' interface, labeled (1), (2), (3), and (4).
 (1) 'SELECIONAR CATEGORIA DE PESQUISA' (Select Search Category). It has three radio buttons: 'RECEITAS' (selected), 'EVENTOS', and 'NOTÍCIAS'.
 (2) 'PESQUISA AVANÇADA - RECEITAS' (Advanced Search - Recipes). It includes:
 - 'GRAU DE DIFICULDADE' (Difficulty Degree) with radio buttons: 'FÁCIL' (selected), 'NORMAL', 'DIFÍCIL'.
 - 'TIPO DE RECEITA' (Recipe Type) with a dropdown arrow.
 - 'INSERIR INGREDIENTE' (Enter Ingredient) with a text input field.
 - 'INSERIR NOME' (Enter Name) with a text input field.
 - 'TEMPO DE PREPARAÇÃO' (Preparation Time) with radio buttons: '0-20m', '20-60m' (selected), '+60m'.
 - A 'PESQUISAR' (Search) button at the bottom.
 (3) 'PESQUISA AVANÇADA - EVENTOS' (Advanced Search - Events). It includes:
 - 'TÍTULO' (Title) with a text input field.
 - 'MÊS (SELECIONAR)' (Month (Select)) with a dropdown arrow.
 - 'LOCALIZAÇÃO' (Location) with a text input field.
 - A 'PESQUISAR' (Search) button at the bottom.
 (4) 'PESQUISA AVANÇADA - NOTÍCIAS' (Advanced Search - News). It includes:
 - 'TÍTULO' (Title) with a text input field.
 - 'MÊS' (Month) with a dropdown arrow.
 - A 'PESQUISAR' (Search) button at the bottom.

Figure 74: Step 5 - "Advanced Search" IS version 1.

Description: The "Advanced Search" IS (Figure 74) was not formally identified in the "Select one of the Choices"-*"Return the Chosen Category"* (Search Situation) *interaction model*. However, it is related to the "Advanced Search" *task*. In this IS the user selects the search category in the "Select Search Category" ("Seleccionar Categoria de Pesquisa") IS (1). Once the category is chosen, the user fills out the search criteria in the correspondent IS:

- "Advanced Search - Recipe" ("Pesquisa Avançada - Receitas")(2) where the user fills out the fields: "Name"("Nome")(d); "Type"("Tipo")(b); "Ingredients" ("Ingrediente")(c); "Difficulty Degree" ("Grau Dificuldade")(a) which can assume the values "Easy"("Fácil"), "Regular"("Normal") and "Hard" ("Difícil"); "Preparation Time"("Tempo Prep.")(e) which can assume the values "0-20" (minutes), "20-60" and "+60";
- "Advanced Search - Events" ("Pesquisa Avançada - Eventos")(3) where the user fills out the fields: "Title"("Título")(f); "Month"("Mês")(g) and "Location" ("Localização")(h);
- "Advanced Search - News" ("Pesquisa Avançada - Notícias")(4) where the user fills out the fields: "Title"("Título")(i) and "Month"("Mês")(j).

The version 1.1 of the IS (Figure 75) is a copy of the version 1. This version was approved without changes which denotes a mistake since the "History" field that was added to the "Recipe" IS clearly should be a search criteria.

Recipes / Events / News

Interaction Space: Recipe Advanced Search

Recipe Name

Recipe Type

Ingredients

Difficulty Level

Preparation Time: 0-20; 20-60; +60.

Search

Interaction Space: Events Advanced Search

Event Title

Month

Location

Search

Interaction Space: News Advanced Search

News Title

Month

Search

Figure 75: Step 5 - "Advanced Search" IS version 1.1.

Following the presentation of the diagrams we now present some conclusions on this work.

V.4. CONCLUSIONS

This chapter presented a case study of the application of the simplified version of *MultiGoals*. In order to test the efficacy of the methodology two Multimedia designers without modeling experience were trained to carry out the analysis and design of the system for a Gastronomy Project.

The analysis and design of the system resulted in: one *use cases model*; three *activity diagrams*; two *interaction models*, and; four *presentation models*. The modeling of the diagrams was made within 3 sessions: (i) training session – 3 persons for 2 hours; (ii) second session – 2 persons for 2 hours; (iii) final modeling session – 3 persons for 1 hour; and the approval session 2 persons for 1 hour. These sessions make a total of 15 hours of modeling including 1 hour of the client.

The modeling resulted in 4 user interfaces that were presented to the client and approved by him, two of them without changes. This can be considered a good result taking into account that the modelers had no previous experience and only received 2 hours of training.

By the analysis of the diagrams produced, we can also conclude that a clarification of the *interaction model* must be done in such a way that the users of the methodology do not confuse it with the *activity diagram*. There also should be a way of relating the *interaction model's* user tasks, system responsibilities and *interaction spaces* in such a way that the resulting diagram is not so complex.

The identification of the user tasks in the *presentation model* contributed for the identification of additional *interaction spaces* that were not previously identified in the *interaction model*. The *interaction model* has contributed to identify additional *system responsibilities* that will play important parts in the system construction.

As a final conclusion, the methodology must be refined following the previous assumptions and be put into practice more times in order to get more feedback for its evolution.

VI. CONCLUSIONS

Goals, including *Process Use Cases* and *MultiGoals* contribute with a set of methods that can be applied to define an Interactive Information System in detail. Indeed, *Process Use Cases* has been applied sufficiently so that it can be considered useful for professional use. *MultiGoals* has already been applied with success in its simplified version and has potential to become a useful tool in the both areas of enterprise IISs and Multimedia.

We believe that the work presented in this thesis is useful in real-life IISs design and that still can be enhanced in the future to achieve optimal results in the discipline of Software Engineering.

VI.1. GENERAL CONCLUSIONS

Goals has been proposed as a solution to organize human resources in order to produce the required artifacts for the requirements, analysis and design phases. *Goals* has proven to be very useful to guide a software development team (this has been done informally at a professional level) through the steps to be taken in order to achieve the definition of a software system.

In particular, for the phases of analysis and design, *Goals* can be considered highly compatible with *Wisdom* [Nuno Nunes, 2001], and *Usage-centered design* [Larry Constantine, 2006]. *Goals* can also be compliant with methodologies such as: (i) *Extreme Programming* (XP) [Kent Beck, 1999] connecting *use cases* with the “user stories” and the *domain model* with the “architectural spike” predicted in XP, and, with (ii) the *Rational Unified Process* (RUP) [Philippe Kruchten, 1999] which provides an extensive set of models to complete the phases of analysis and design. As an extra requirement, the compatibility of the definitions of: *essential use case* (*use case*) [Larry Constantine, 2006], *entity* (set of information) [Nuno Nunes, 2001] and *actor* should also be observed.

We discuss on the next sections some specific conclusions related to the phases of requirements (*Process Use Cases*), analysis and design (*MultiGoals*).

VI.1.1. Requirements (Process Use Cases)

Process Use Cases (PUC) is a methodology that identifies *use cases* as a leap for software construction producing valid artifacts for both activities of Business Process Management (BPM) and Software Engineering (SE). The identification of business processes and their design can be used for BPM activities, while the *domain model* and the identified *use cases* can be used for SE activities.

In a modeling perspective, achieving the correct level of abstraction to name *use cases* can be a very difficult task in SE if no global comprehension exists of the scope of the project within the enterprise organization. Using PUC it is easier to reach the appropriate abstraction to nominate the (*essential*) *use cases* in a way that they make sense in both BPM and SE disciplines.

PUC is distinct from the other approaches in the way that: (i) it makes the reorganization of the business towards automation more elucidative to users, since BPs and *use cases* are designed in a single model that can be understood by every stakeholder; (ii) it includes an information-oriented strategy that enables the selection of the BPs that really need to be designed in order to achieve automation, and; (iii) it is oriented to software development, once both *use cases* and information *entities* are already identified when PUC is finished.

Table 6: Requirements methodologies comparison.

	PUC	Dijkman	Gonzales	Štolfa	NA	Wisdom	UCD
Project Interiorization	X					X	
Business Strategy Description			X				
Data Modeling	X		2			X	3
Business Process Context	X					X	X
Goals Identification	X		X				
Business Process Design	X	X	X	X	X	X	X
Business Rules					X		
Business Processes Resources	1		X				X
Use Cases Identification	X	X	X	X	X	X	X

1 Only Information Resources

2 Domain model of the used resources

3 Systems, Artifacts and Tools Identification

By the analysis of Table 6, we can conclude that PUC only lacks the Business Strategy Description and the definition of the Business Rules. However, PUC can be enhanced in the future to cover the missing issues as discussed on the next section.

PUC has already been applied within over 10 different real software development projects for the Information and Computing Center at University of Madeira (UMa), Portugal, for the automation of at least one *business process* per project. It was applied by both undergraduate students and IT professionals, and it was shared with UMa managers always resulting in a consistent artifact that promoted consensus among the stakeholders.

VI.1.2. Analysis and Design (MultiGoals)

MultiGoals is a methodology that aims to bridge SE and Multimedia authoring. It is a lightweight, *use case*-driven, architectural centric methodology that guides the software conceptualization by means of the simplification of the system design concerning usability and maintainability.

The use of *MultiGoals* induces the author of an IIS into a straight lined (few iterations are suggested) and fast software definition process towards implementation. The models of the methodology are intended to be simple, intuitive and scalable.

The development of *MultiGoals* has proven to be a very difficult task once all requirements of both regular SE applications and IMDs had to be taken into account. The full definition of the system behavior could only be reached by the description of a set of patterns that are applied to define the structure of *system responsibilities* and domain classes (or objects) that support the system functionalities.

Despite the proposal of 11 steps for the design of Interactive Information Systems, these steps support completely the design of complex applications. However, as a matter of simplicity, many of these steps can be omitted in order to rapidly achieve the products of the methodology. In the simplified version of *MultiGoals* only 4 models are applied and satisfactory results can be achieved, as presented in Chapter V - Case Study of *MultiGoals*.

When comparing the *MultiGoals* methodology with the other approaches, we can conclude that it is a more balanced and complete methodology, since it considers all modeling aspects from user requirements definition to Multimedia specific implementation problems. *MultiGoals* is a structured methodology that allows the modeling of all the IISs tiers taking into account the user interaction and navigation with support for Multimedia.

Furthermore, a special attention is given to the maintenance and reusability of the IIS where each component of the system: *interaction space* (user interface), *task*, *system responsibility* or *entity*, assumes a relevant and well defined role. *MultiGoals* was proposed to offer user friendly diagrams supporting different components of an IIS. In particular, this characteristic enables the traceability of the IIS design.

Table 7: Analysis and Design methodologies comparison.

	MultiGoals	UWE	W2000	OMMMA
Use Cases	X	X	X	
Interface Design	X	X		X
Navigation	X	X	X	
Interaction	X	X	X	X
System Responsibilities	X		1	
Synchronization	X			X
Database Modeling	X	X	X	
Content Modeling	X			X

1 Rules are defined in the use cases model for the access to the information that will be later implemented by the system responsibilities.

By the analysis of Table 7, we can conclude that *MultiGoals* covers all the defined criteria integrating in a single methodology the contribution of both Hypermedia-oriented (UWE and W2000) and IMD-oriented (OMMMA) methodologies.

The first application of the simplified version of *MultiGoals* (Chapter V - Case Study) has proven that the methodology is easy to use even for modelers with no experience

and can produce acceptable results. However, the remaining steps of the methodology have only been applied academically for the conception of small examples including those presented in this thesis.

VI.2. FUTURE WORK

Some general enhancements can still be proposed for *Goals*, such as:

- **Completion of the SE lifecycle phases**

Goals should be completed in order to cover the remaining phases of development, test, installation (and maintenance). Some clues for the development phase can be the code generation from the modeled classes as a boost for software implementation. As for the testing phase, it could be based on a black-box approach following the defined user *tasks*.

- **Quality of the Software**

The definition of software quality attributes could represent a major advance on the requirements which are not captured by *use cases*. Examples of this situation are availability and performance. Some of the software quality attributes are defined at a global level like Cost and Schedule, others, like performance would be defined by the chosen methodologies for requirements, analysis and design.

- **Project Management**

The definition of solutions for the parallel activity of project management would be a major improvement. Issues like scheduling, viability, risk management, and team work directives can be very useful for large projects.

Some specific enhancements can also be proposed for PUC and *MultiGoals*, as follows.

VI.2.1. Requirements (Process Use Cases)

Process Use Cases could benefit from the following enhancements:

- **Efficiency and Efficacy Measurement**

As already mentioned in Chapter III.2. (*Process Use Cases: An Overview*), PUC does not cover the phase of monitoring BPs of a Business Process Management lifecycle. For this purpose, it would be interesting to analyze the flow of the business process measuring efficacy and efficiency of each *task* to easily identify bottlenecks and, consequently, reorganize the BPs based on measured evidences.

- **Implementation Effort Estimation**

Important decisions must be taken at the end of the phase of requirements in a Software Engineering lifecycle, normally related to the cost of implementation of the identified requirements, and related to the decision of what to be implement first. If in one hand it is easy to know what is most valuable to the client, on the other hand it is really difficult to know what will be the “price” of the requirements.

However, there are some indicators that can provide valuable information for this purpose, even if with relatively big margin of error, these indicators are: estimation of the number of information entities manipulated by the *use case*, number of records of the manipulated information entities, number of users related to the *use case*.

- **Business Strategy**

Business strategy is closely related to business goals. PUC already establishes a direct connection between business processes and business goals. The establishment of a relation to the business strategy in a top-down approach would be a major advancement in order to ensure the alignment between business and information system. Business strategy allied to the effort estimation could ease the decision for the activity of prioritizing the implementation of the software.

Proposals already exist that model strategy and business goals, such as [André Vasconcelos et al., 2001] which is based on the well known Balanced Scorecard [The Balanced Scorecard Institute, 2007].

- **Business Rules**

The capture of business rules is an important feature regarding the implementation of an Interactive Information System, since these business rules have to be implemented within the *system responsibilities*. During the design of the business processes, an additional effort could be made in order to identify these rules in advance, also providing a good contribution for implementation effort estimation. The identification of the consumption and production of physical resources could also be related to the business rules.

VI.2.2. Analysis and Design (MultiGoals)

MultiGoals could benefit from the following enhancements:

- **QoE and QoS Measurement**

The Quality of Experience (QoE) of an application is the degree of satisfaction related to the experience the user has interacting with the used application. QoE is difficult to measure, however it is our belief that usability can be measured based on the try-and-

error attempts made by the user to successfully achieve the completion of a complete and meaningful *task*. An usability coefficient would provide an important indicator for QoE measurement.

Especially concerning Multimedia, Quality of Service (QoS) indexes can be associated with each *multimedia system responsibility* in order to provide the application with an indicator of the priorities (related to each media) that need to be guaranteed. These QoS requirements could be introduced during the modeling of the system, since the author has a general and privileged perspective of all the components involved in it.

- **CASE Tool**

The implementation of a CASE tool would definitely be a major improvement. This would involve the construction of an integrated environment including the modeling tools and the authoring tools for Multimedia. A solution would possibly be the integration of the models in a tool like metaSketch [Leonel Nóbrega, 2007].

Meanwhile, the authoring of applications using *MultiGoals* can also be accomplished using any UML 2.0 [Object Management Group, 2003] compliant CASE tool. In this work, special attention must be taken to usability, regarding the integration of both system and media information in order to achieve self-explained applications.

Diagrams Improvement

As a result from the analysis of the models of the first application of the *MultiGoals* simplified version, it was concluded that the *interaction model* must be improved in order to produce the same results with a less complex and confusing diagram. The same approach must be used to the complete version of the methodology in order to be able to obtain propose solutions for the evolution of *MultiGoals*.

- **Patterns Improvement**

The number of patterns should be enlarged in order to support more system behaviors, and as a complement it also would be a major improvement the implementation of a system of validation for the used patterns, possibly integrated in a CASE tool.

This thesis presented three approaches that have as purpose the professional use in the area of Software Engineering. It is our belief that our work has contributed with one more and important step to close the existing gap between traditional Interactive Information Systems and Interactive Multimedia Documents modeling.

LIST OF PUBLICATIONS

Pedro Valente, Paulo Sampaio. (2006) *Defining Goals for the design of Interactive Multimedia Documents*. In P. Kommers & G. Richards (Eds.), *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*. pp. 955-962. June 2006, Chesapeake, VA: AACE.

Pedro Valente, Paulo Sampaio. (2007). *Goals: Interactive Multimedia Documents Modeling*. In K. Luyten (Ed.), *Lecture Notes in Computer Science*. Vol. Volume 4385/2007, pp. 169-185. Hasselt, Belgium: Springer Berlin / Heidelberg. ISBN: 978-3-540-70815-5.

Pedro Valente, Paulo Sampaio. (2007). *Process Use Cases: Use Cases Identification*. In *Proceedings of ICEIS'2007 - 9th International Conference on Enterprise Information Systems*, Funchal, Madeira, Portugal. Vol. *Information Systems Analysis and Specification*, pp. 301-307.

REFERENCES

- [Adobe, 2007a] Adobe. (2007a). Adobe Flash CS3 Professional. Retrieved December 19th 2007: <http://www.adobe.com/products/flash/>
- [Adobe, 2007b] Adobe. (2007b). Adobe Photoshop CS3. Retrieved December 19th 2007: <http://www.adobe.com/br/products/photoshop/photoshop/>
- [Alberto Silva and Carlos Videira, 2005] Alberto Silva, Carlos Videira. (2005). *UML, Metodologias e Ferramentas CASE* (2^a ed. Vol. 1). Centro Atlântico. ISBN: 989-615-009-5.
- [Alejandro Jaimes et al., 2006] Alejandro Jaimes, Nicu Sebe, Daniel Gatica-Perez. (2006, Aug. 2006). *Human-Centered Computing: A Multimedia Perspective*. In Proceedings of ACM. International Conference on Multimedia, USA. pp. 855-864.
- [André Vasconcelos et al., 2001] André Vasconcelos, Artur Caetano, João Neves, Pedro Sinogas, Ricardo Mendes, José Tribolet. (2001). *A Framework for Modeling Strategy, Business Processes and Information Systems*. In Proceedings of Enterprise Distributed Object Computing Conference (EDOC). pp. 69-80.
- [Athula Ginige et al., 1995] Athula Ginige, David Lowe, John Robertson. (1995). Hypermedia Authoring. *IEEE Multimedia*. Vol. 2(4).
- [Autodesk, 2007] Autodesk. (2007). 3D Studio Max. Retrieved December 19th 2007: www.autodesk.com/3dsmax
- [Bashar Nuseibeh and Steve Easterbrook, 2000] Bashar Nuseibeh, Steve Easterbrook. (2000). *Requirements Engineering: A Roadmap*. In Proceedings of International Conference on Software Engineering (ICSE-2000), Limerick, Ireland. pp. 35-46.
- [Boris Shishkov and Jan Dietz, 2005] Boris Shishkov, Jan Dietz. (2005). Deriving Use Cases from Business Processes. In S. Netherlands (Ed.), *Enterprise Information Systems V*. Vol. Computer Science, pp. 249-257. ISBN: 978-1-4020-1726-1 (Print) 978-1-4020-2673-7 (Online).
- [Charles Kreitzberg, 1999] Charles Kreitzberg. (1999). *The LUCID Framework (Logical User Centered Interaction Design) (Pre-Release Version 0.4)*. Retrieved December 19th 2007: <http://www.cognetics.com>
- [Christopher Alexander, 1979] Christopher Alexander. (1979). *The Timeless Way of Building*. Oxford University Press. ISBN: 978-0195024029.

- [Chung-Ming Huang et al., 2004] Chung-Ming Huang, Jyh-Shiou Chen, Chih-Hao Lin, Chian Wang, Chung-Ming Huang, Jyh-Shiou Chen, Chih-Hao Lin, Chian Wang. (2004). MING-I: a distributed interactive multimedia document development mechanism. *Multimedia Systems*. Volume 6(Number 5 / September, 1998), pp. 316-333. ISSN: 0942-4962 (Print) 1432-1882 (Online).
- [Daniel Schwabe and Gustavo Rossi, 1998] Daniel Schwabe, Gustavo Rossi. (1998). *Developing Hypermedia Applications using OOHDM*. In Proceedings of Workshop on Hypermedia Development Process, Methods and Models. Hypertext'98, Pittsburg, USA
- [Dave Robert et al., 1998] Dave Robert, Dick Berry, Scott Isensee, John Mullaly. (1998). *Designing for the user with OVID : bridging user interface design and software engineering* (Lst Ed edition (September 17, 1998) ed.). Macmillan Technical Pub. ISBN: 978-1578701018.
- [Donald Chamberlin and Raymond Boyce, 1974] Donald Chamberlin, Raymond Boyce. (1974). *Structured English Query Language*. In Proceedings of International Conference on Management of Data Archive, Ann Arbor, Michigan. pp. 249 - 264.
- [Edward Yourdon and Larry Constantine, 1979] Edward Yourdon, Larry Constantine. (1979). *Structured design : fundamentals of a discipline of computer program and systems design*. Prentice Hall. ISBN: 978-0138544713.
- [Fábio Paternò et al., 1997] Fábio Paternò, Cristiano Mancini, Silvia Meniconi. (1997). *ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models*. In Proceedings of INTERACT '97, Proceedings of the IFIP TC13 International Conference on HCI.362-369.
- [Filipe Freitas, 2007] Filipe Freitas. (2007). Masilli. Retrieved December 19th 2007: www.masilli.com
- [Franca Garzotto et al., 1993] Franca Garzotto, Paolo Paolini, Daniel Schwabe. (1993). HDM—a model-based approach to hypertext application design. *ACM Transactions on Information Systems (TOIS)*. Vol. 11(1), pp. 1-26. ISSN: 1046-8188.
- [Hans-Erik Eriksson and Magnus Pencker, 2001] Hans-Erik Eriksson, Magnus Pencker. (2001). *Business Modeling With UML: Business Patterns at Work* (1st edition ed.). John Wiley & Sons. ISBN: 0471295515.
- [Ian Sommerville, 2005] Ian Sommerville. (2005). Integrated Requirements Engineering: A Tutorial. *IEEE Software archive*. Vol. 22(1), pp. 16 - 23. ISSN: 0740-7459.
- [International Standards Organization, 1999] International Standards Organization. (1999). *ISO 13407:1999. Human-centred design processes for interactive systems. First edition*.
- [James Rumbaugh et al., 1999] James Rumbaugh, Ivar Jacobson, Grady Booch. (1999). *The Unified Modeling Language Reference Manual*. Addison-Wesley Professional. ISBN: 978-0201309980.
- [Jana Koehler et al., 2002] Jana Koehler, Giuliano Tirenni, Santhosh Kumaran. (2002, 17-20 September 2002). *From Business Process Model to Consistent Implementation: A Case for Formal Verification Methods*. In Proceedings of IEEE International Enterprise Distributed Object Computing Conference (EDOC), Lausanne, Switzerland. pp. 96-106.
- [Jose González and Juan Sánchez Díaz, 2007] Jose González, Juan Sánchez Díaz. (2007, 11-15 June 2007,). *Business process-driven requirements engineering: a goal-based approach*. In Proceedings of 8th Workshop on Business Process Modeling, Development, and Support (BPMDS'07), Trondheim, Norway.

- [Kent Beck and Ward Cunningham, 1989] Kent Beck, Ward Cunningham. (1989, October 1-6). *A Laboratory For Teaching Object-Oriented Thinking*. In Proceedings of Object-Oriented Programming, Systems, Languages, and Applications 89, New Orleans, Louisiana. Vol. 24, pp. 1-6.
- [Kent Beck, 1999] Kent Beck. (1999). *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional; US Ed edition. ISBN: 978-0201616415.
- [Khalil Mehdi El-Khatib, 2005] Khalil Mehdi El-Khatib. (2005). *A QoS Content Adaptation Framework for Nomadic Users*. Phd Thesis. School of Information Technology and Engineering University of Ottawa, Ottawa, Ontario, Canada.
- [Kyoungro Yoon and Bruce Berra, 1998] Kyoungro Yoon, Bruce Berra. (1998, 5-7 August). *TOCPN: interactive temporal model for interactive multimedia documents*. In Proceedings of International Workshop on Multi-Media Database Management Systems. pp. 136 - 144.
- [Larry Constantine and Lucy Lockwood, 2000] Larry Constantine, Lucy Lockwood. (2000). Structure and Style in Use Cases for User Interface Design. In *Object Modeling and User Interface Design*. Boston: Addison Wesley. ISBN: 978-0201657890.
- [Larry Constantine, 2002] Larry Constantine. (2002). Usage-Centered Engineering for Web Applications. *IEEE Software*. Vol. 19(2), pp. 42-50.
- [Larry Constantine, 2003] Larry Constantine. (2003). Canonical Abstract Prototypes for Abstract Visual and Interaction Design. *LNCS - Lecture Notes in Computer Science*, pp. 1-15. ISSN: 0302-9743 (Print) 1611-3349 (Online).
- [Larry Constantine, 2006] Larry Constantine. (2006). *Activity Modeling: Toward a Pragmatic Integration of Activity Theory with Usage-Centered Design*. LabUSE, Universidade da Madeira. Retrieved December 19th 2007:
<http://www.foruse.com/articles/activitymodeling.htm>.
- [Leonel Nóbrega, 2007] Leonel Nóbrega. (2007). metaSketch. Retrieved December 19th 2007:
http://dme.uma.pt/labuse/?page_id=4
- [Luciano Baresi et al., 2001] Luciano Baresi, Franca Garzotto, Paolo Paolini. (2001). *Extending UML for Modeling Web Applications*. In Proceedings of 34th Hawaii International Conference on System Sciences, Maui, Hawaii. Vol. 3.
- [Maria Escalona et al., 2003] Maria Escalona, Manuel Mejías, Jesús Torres, Antonia Reina. (2003). The NDT Development Process. *Lecture Notes in Computer Science*. Vol. 2722/2003, pp. 19-25. ISSN: 0302-9743 (Print) 1611-3349 (Online).
- [Mark Van Harmelen, 2001] Mark Van Harmelen. (2001). Designing with Idiom. In M. v. Harmelen (Ed.), *Object Modeling and User Interface Design*. pp. 71-113: Addison-Wesley. ISBN: 978-0201657890.
- [Nora Koch and Andreas Kraus, 2002] Nora Koch, Andreas Kraus. (2002). *The Expressive Power of UML-based Web Engineering*. In Proceedings of International Workshop on Web-Oriented Software Technology (IWOST'02), Málaga, Spain. pp. 105-119.
- [Nuno Nunes, 2001] Nuno Nunes. (2001). *Object Modeling for User-Centered Development and User Interface Design: The Wisdom Approach*. Phd Thesis. Universidade da Madeira.
- [Object Management Group, 2003] Object Management Group. (2003). *Unified modeling language superstructure, version 2.0. final adopted specification*. Retrieved December 19th 2007:
<http://www.omg.org/docs/formal/05-07-04.pdf>.

- [Paulo Vieira, 2007] Paulo Vieira. (2007). Buscarov. Retrieved December 19th 2007: <http://www.buscarov.com/>
- [Pedro Valente and Paulo Sampaio, 2007a] Pedro Valente, Paulo Sampaio. (2007a). Goals: Interactive Multimedia Documents Modeling. In K. Luyten (Ed.), *Lecture Notes in Computer Science*. Volume 4385/2007, pp. 169-185. Hasselt, Belgium: Springer Berlin / Heidelberg. ISBN: 978-3-540-70815-5.
- [Pedro Valente and Paulo Sampaio, 2007b] Pedro Valente, Paulo Sampaio. (2007b). *Process Use Cases: Use Cases Identification* In Proceedings of ICEIS'2007 - 9th International Conference on Enterprise Information Systems, Funchal, Madeira, Portugal. Vol. Information Systems Analysis and Specification, pp. 301-307.
- [Philippe Kruchten, 1999] Philippe Kruchten. (1999). *The Rational Unified Process (An Introduction)*. Addison-Wesley Professional. ISBN: 978-0201707106.
- [Remco Dijkman and Stef Joosten, 2002] Remco Dijkman, Stef Joosten. (2002). *Deriving Use Case Diagrams from Business Process Models*. University of Twente.
- [Rikard Land, 2002] Rikard LandRikard Land. (2002). A Brief Survey of Software Architecture. *MRTC Technical Report*. ISSN: 1404-3041.
- [Robert Bretl et al., 1999] Robert Bretl, Allen Otis, Marc San Soucie, Bruce Schuchardt, R. Venkatesh. (1999). *Persistent Java Objects in 3 tier architectures*. In Proceedings of 3rd International Workshop on Persistence and Java (PJW3): Advances in Persistent Object Systems. pp. 236-249.
- [Roberto Willrich, 1996] Roberto Willrich. (1996). *Conception formelle de documents hypermedias portables*. Phd Thesis. Paul Sabatier, Toulouse, France.
- [Stefan Sauer and Gregor Engels, 2001] Stefan Sauer, Gregor Engels. (2001). *UML-based Behavior Specification of Interactive Multimedia Applications*. In Proceedings of IEEE Int'l Symposium on Human-Centric Computing Languages and Environments (HCC), Stresa, Italy248--255.
- [Stefano Ceri et al., 2003] Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, Maristella Matera. (2003). *Designing Data-Intensive Web Applications*. Morgan Kaufmann; 1 edition. ISBN: 978-1558608436.
- [Susanne Boll and Wolfgang Klas, 2001] Susanne Boll, Wolfgang Klas. (2001). *ZYX - A Multimedia Document Model for Reuse and Adaptation*. In Proceedings of IEEE Transactions on Knowledge and Data Engineering. Vol. 3, pp. 361-382.
- [Svatopluk Štolfa and Ivo Vondrák, 2006] Svatopluk Štolfa, Ivo Vondrák. (2006). *Mapping from Business Processes to Requirements Specification*. Universitat Trier.
- [The Balanced Scorecard Institute, 2007] The Balanced Scorecard Institute. (2007). Balanced Scorecard. Retrieved December 19th 2007: <http://www.balancedscorecard.org/>
- [The World Wide Web Consortium, 2007] The World Wide Web Consortium. (2007). Synchronized Multimedia Integration Language 2.0. Retrieved December 19th 2007: <http://www.w3.org/AudioVideo/>
- [Tom Dayton et al., 1998] Tom Dayton, Al Mcfarland, Joseph Kramer. (1998). Bridging User Needs to Object Oriented GUI Prototype via Task Object Design. In L. Wood (Ed.), *User Interface Design: Bridging the Gap from Requirements to Design*. pp. 15-56. ISBN: 0-8493-3125-00.

[Webinterx, 2006] Webinterx. (2006). Services. Retrieved December 19th 2007:
<http://www.webinterx.com/services/index.html>

APPENDIX A: STATIC AND RUN-TIME PATTERNS

The *MultiGoals* patterns are a solution to model system behavior by means of the definition of a structure of *system responsibilities* (or alternatively *interaction spaces*), classes and/or objects.

Each *system responsibility* has a “(system responsibility) name” that defines which pattern is being used. The composition of the “system responsibility name” is based on “Keywords” that are combined with references to attributes of classes (e.g. “attribute name”) and/or classes (e.g. “class name”). The “system responsibility name” is concatenated by means of spaces (“ ”).

The pattern structure has an associated meaning in terms of the functioning of the system which is defined by its “purpose”.

Static Patterns

The static patterns define a structure composed by a *system responsibility* that has an action over a source, which is a class or a structure of classes from the *application domain model*.

Get/Set Class Attribute

PURPOSE: when the objective is to read or change the value of an attribute of a class, the user *tasks* associated with an *interaction space* (where the attribute is manipulated) can be associated with a *system responsibility*, and that respective *system responsibility* associated with a class from the *ontological domain model* where the attribute exists.

SYSTEM RESPONSIBILITY NAME: [keyword: GET (or SET, or GET/SET)]+[]+[Class Name]+[]+[Attribute Name]

SOURCE: Class from the *ontological domain model*.



Figure 76: Get/Set Class Attribute pattern.

In the example presented in figure 76, the *system responsibility* will perform an action (read or write) over an attribute (defined in the *system responsibility* name) of the associated class.

Get/Set Correspondent Media

PURPOSE: when the objective is to read (Get) or change (Set) the value of a media (from the *media domain model*) associated with a class from the *ontological domain model*, the user *tasks* associated with an *interaction space* (where the media is manipulated) can be associated with a *system responsibility*, and that respective *system responsibility* associated with a class from the *ontological domain model* where the media exists.

SYSTEM RESPONSIBILITY NAME: [keyword: GET (or SET, or GET/SET)] +[]+[keyword: CORRESPONDENT] +[]+[Class Name]+[]+[Media Class Name]

SOURCE: Class from the *media domain model*.

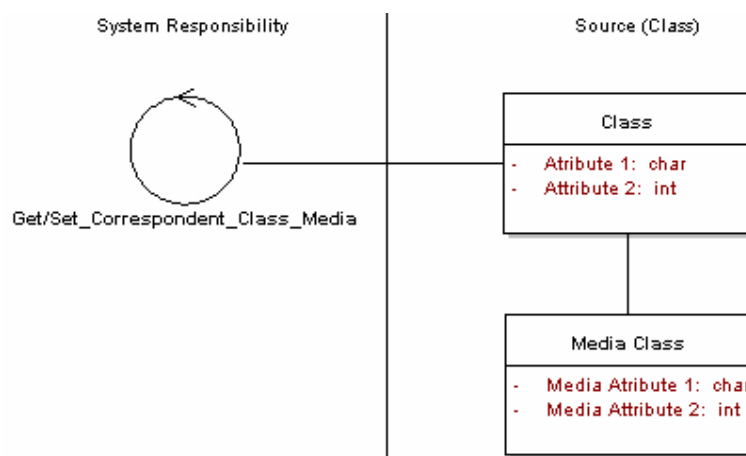


Figure 77: Get/Set Correspondent Media pattern.

In the example presented in figure 77, the *system responsibility* will perform an action (read or write) over a media class (defined in the *system responsibility* name) that has a relation of association with the class that the *system responsibility* is associated with.

Action Over Correspondent Media

PURPOSE: when the objective is to perform/execute an action (e.g. Present or Interrupt) over a media (from the *media domain model*) associated with a class from the *ontological domain model*, the user tasks associated with an *interaction space* (where the media is manipulated) can be associated with a *system responsibility*, and that *system responsibility* associated with a class from the *media domain model* where the media exists.

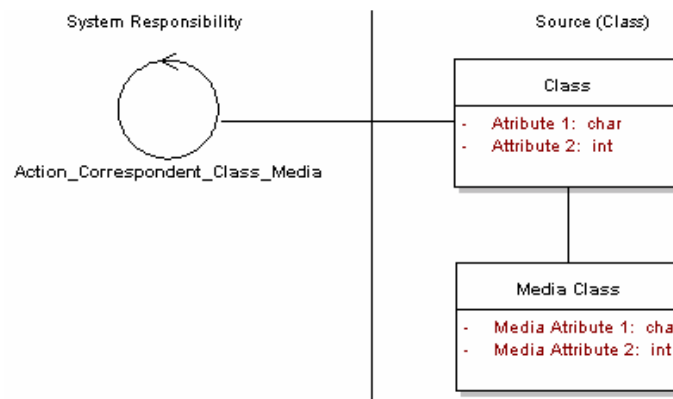


Figure 78: Action Correspondent Media pattern.

SYSTEM RESPONSIBILITY NAME: [Action Name] +[]+[keyword: CORRESPONDENT] +[]+[Class Name]+[]+[Media Class Name]

SOURCE: Class from the *media domain model*.

In the example presented in figure 78, the *system responsibility* will perform an action (for example present or interrupt) over a media class (defined in the *system responsibility* name) that has a relation with the class that the *system responsibility* is associated with.

Run-Time Patterns

The run-time patterns define a structure composed by an *interaction space* or *system responsibility* that has a relation with a source, which is one or more objects from the *application object model*.

Source Region (No System Responsibility and no Source)

PURPOSE: If an *interaction space* has no associated *system responsibilities*, a two-tier association can be made from the *interaction space* to the correspondent object instantiated with default values from the *region object model*.

SYSTEM RESPONSIBILITY NAME: N/A¹

SOURCE: Object from the *region object model*.

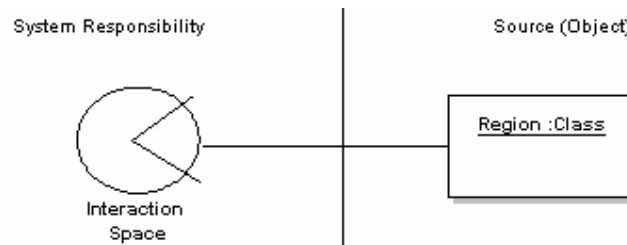


Figure 79: Source Region pattern

In the example presented in Figure 79 the object provides the default values for the *interaction space*.

Source Media

PURPOSE: If an *interaction space* has no associated *system responsibilities* but has a media as a static source, a two-tier association can be made from the *interaction space* to the correspondent object instantiated with default values from the *media object model*.

SYSTEM RESPONSIBILITY NAME: N/A

SOURCE: Object from the *media object model* (media).

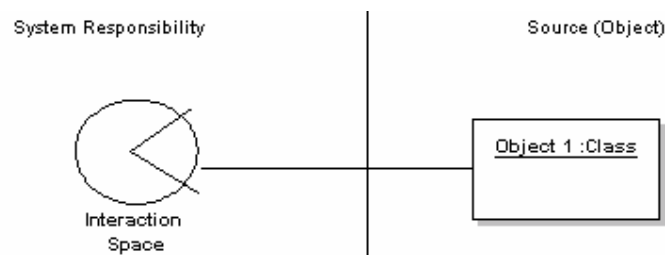


Figure 80: Source Media pattern

In the example presented in Figure 80 the object provides the media for the *interaction space*.

¹ Not Applicable

Exclusive (system responsibility)

PURPOSE: If an *interaction space* has an associated *system responsibility* that has more than one media object as a dynamic source in a condition of an exclusive presentation (all media are executed but never simultaneously), a three-tier association can be made from the *interaction space* to the correspondent objects instantiated with default values from the *media object model*.

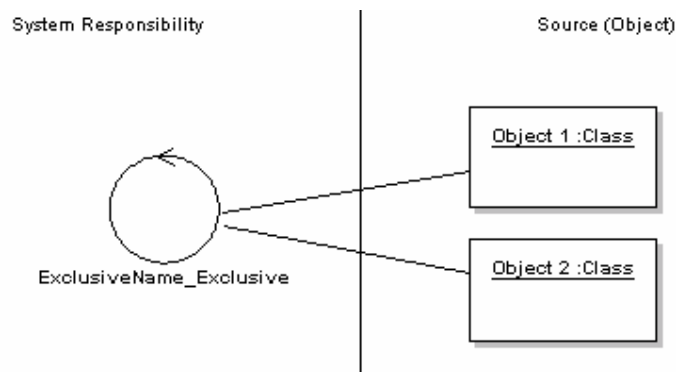


Figure 81: Exclusive pattern

SYSTEM RESPONSIBILITY NAME: [Exclusive Name]+[]+[keyword: EXCLUSIVE]

SOURCE: Objects instantiated preferably with default values from the *media object model*.

In the example presented in Figure 81 a *system responsibility* and two media objects define a structure of an exclusive presentation.

Parallel (system responsibility)

PURPOSE: If an *interaction space* has an associated *system responsibility* that has more than one media object as a dynamic source in a condition of a parallel presentation (all media are executed at the same time), a three-tier association can be made from the *interaction space* to the correspondent objects instantiated with default values from the *media object model*.

SYSTEM RESPONSIBILITY NAME: [Parallel Name]+[]+[keyword: PARALLEL]

SOURCE: Objects instantiated preferably with default values from the *media object model*.

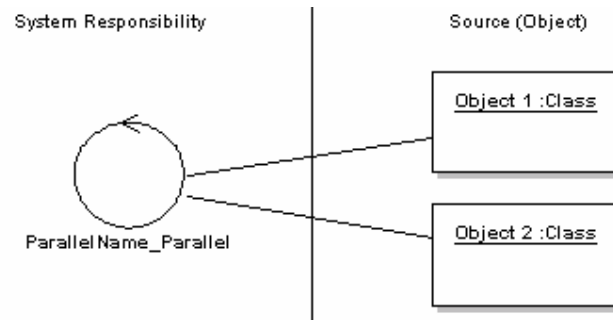


Figure 82: Parallel pattern

In the example presented in Figure 82 a *system responsibility* and two media objects define a structure of a parallel presentation.

Sequence (system responsibility)

PURPOSE: If an *interaction space* has an associated *system responsibility* that has more than one media object as a dynamic source in a condition of sequence (one media after another), a three-tier association can be made from the *interaction space* to the correspondent objects instantiated with default values from the *media object model*.

SYSTEM RESPONSIBILITY NAME: [Sequence Name]+[]+[keyword: SEQUENCE]

SOURCE: Objects instantiated preferably with default values from the *media object model*.

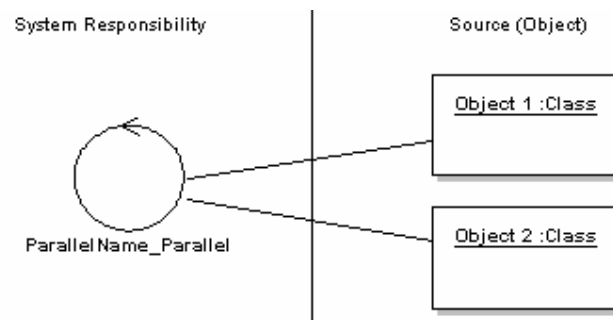


Figure 83: Sequence pattern

In the example presented in Figure 83 a *system responsibility* and two media objects define a structure of sequence.

APPENDIX B: STEREOTYPES

This section presents the meaning of the stereotypes used along the modeling of both *Process Use Cases* and *MultiGoals* methodologies.

Actor

An *actor* is someone who carries out a *task*/activity within a business process (in interaction or not with the system).

Goal

A *goal* is the final product of a business process.

Business Process

A business process is a sequence of activities/*tasks* carried out by *actors* in order to achieve an enterprise *goal*.

Use Case

An *use case* is a top-level *task* carried out by a user in a business process that is understandable by every stakeholder of the Interactive Information System as a complete action. This definition is complemented and completed with the definition of *use case* provided by Larry Constantine and Lucy Lockwood in [Larry Constantine and Lucy Lockwood, 2000] as previously presented in section II.4.4. Analysis and Design Base Techniques -> Essential Use Cases.

Task

In *MultiGoals* the definition of *task* follows the definition provided within *Wisdom* [Nuno Nunes, 2001]: “*Task* classes are used to model the structure of the dialogue between the *user* and the system in terms of meaningful and complete sets of actions required to achieve a goal. *Task* classes are responsible for *task* level sequencing, consistency of multiple presentation elements and mapping back and forth between entities and presentation classes (*interaction spaces*). *Task* classes encapsulate the complex temporal dependencies and other restrictions among different activities required to use the system and that cannot be related to specific entity classes. Thereby, *task* classes isolate changes in the dialogue structure of the user interface.”

This definition is compatible with the definition provided by Fábio Paternò in *ConcurTaskTrees* [Fábio Paternò et al., 1997] except that only *tasks* carried out by the user are used in *MultiGoals* (excluding the application type of *task* provided by the same document which is carried out by the system, defined in *MultiGoals* as a *system responsibility*): “A *task* defines how the user can reach a goal in a specific application domain. The goal is a desired modification of the state of a system or a query to it.”

The *audio task* is a particular case of *task* in which the dialogue between user and system is made by audio.

Activity

An activity is a top-level *task* performed by a user with no interaction with a system.

Interaction Space

In *MultiGoals* the definition of *interaction space* follows the definition provided within *Wisdom* [Nuno Nunes, 2001]: “The *interaction space* class is used to model interaction between the system and the human users. An *interaction space* class represents the space within the user interface of a system where the user interacts with all the functions, containers, and information needed to carry out some particular *task* or set of interrelated *tasks*. *Interaction space* classes are responsible for the physical interaction with the *user*, including a set of interaction techniques that define the image of the system (output) and the handling of events produced by the user (input). *Interaction space* classes isolate change in the user interface of the system, *interaction spaces* are technology independent, nevertheless they often represent abstraction of windows, forms, panes, etc.”

The *audio interaction space* is a particular case of *interaction space* in which both input and output are made by audio.

Control (System Responsibility)

In *MultiGoals* the definition of *control* follows the definition provided within *Wisdom* [Nuno Nunes, 2001]: “The *control* class represents coordination, sequencing, transactions and control of other objects. Control classes often encapsulate complex derivations and calculations (such as business logic) that cannot be related to specific entity classes. Thereby, control classes isolate changes to control, sequencing, transactions and business logic that involves several other objects.”

Entity

In *MultiGoals* the definition of *control* follows the definition provided within *Wisdom* [Nuno Nunes, 2001]: “The entity class is used to model perdurable information (often persistent). Entity classes structure domain classes and associate behavior, often, representing a logical data structure. As a result, entity classes reflect the information in a way that benefits developers when designing and implementing the system (including support for persistence). Entity objects isolate changes to the information they represent.”

Class

A class represents a discrete concept within the application being modeled [James Rumbaugh et al., 1999]. A class is the descriptor for a set of objects with similar structure, behavior, and relationships that have state and behavior. The state is described by attributes and associations. Attributes are used for pure data values such as numbers and strings. Individual pieces of invocable behavior are described by operations; a method is the implementation of an operation.

Object

An object is an instance of a class [James Rumbaugh et al., 1999]. An instance is a run-time entity with identity, that is, something that can be distinguished from other run-time entities. An object has one data value for each attribute in its corresponding class.

Association

Association defines a relationship between classes of objects [Alberto Silva and Carlos Videira, 2005]. It is a semantic relation among two elements of a model. In a class diagram, an association defines the rules that establish and guarantee the integrity of the relation among objects of the participating classes. This includes: the relation name; the number of objects that can take part of the association.

Generalization

Generalization is a relation between a general element (the super-class) and a specific element (sub-class) [Alberto Silva and Carlos Videira, 2005]. Generalization is a relation of the type “is a kind of”.

Aggregation and Composition

The aggregation association is a relation of the kind “is part of” that corresponds to the fact that an instance of a given class is composed by one or more instances of another class [Alberto Silva and Carlos Videira, 2005].

Composition, also called “strong aggregation” is a variant of the aggregation association. It means that the sub-class can not exist without the existence of the super-class.

Usage

Usage is a relation of dependency and reflects a relation of the type client-supplier, where a change in the supplier element means a change in the client element, but the contrary is not necessarily true [Alberto Silva and Carlos Videira, 2005].

Transition

A transition leaving a state defines the response of an object in the state to the occurrence of an event. In general, a transition has an event (usually the completion of an activity), a guard condition, an action, and a target state.